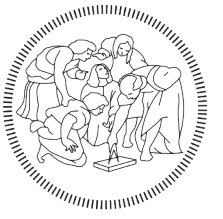
POLITECNICO DI MILANO
Master of Science in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria



POLITECNICO
MILANO 1863

Identifying binge-watching behavior on serial content and exploring its relevance in Recommender Systems

Supervisor: Paolo Cremonesi
Co-supervisors: Mario Scriminaci
Fernando B. Pérez Maurera

M.Sc. Thesis by:
Matteo Carretta, matriculation number 922378

Academic Year 2019-2020

To Astrid, my soulmate, forever

Abstract

With the growth of Over-The-Top media services, the way content is delivered to viewers changed dramatically. On such platforms, every episode of TV-series seasons is released most of the time simultaneously. Consequently, viewers' behavior has evolved, and they often watch several episodes one after the other in most of their watching sessions. This phenomenon is referred to as binge-watching, while binge-worthy series are series prone to be binge-watched. Therefore, this thesis aims to provide methods to identify binge-watchers and binge-worthy series and explore the relevance of this information as features of Recommender Systems.

First, we identify user-series watching sessions, and we extract binge-watchers and binge-worthy series by applying high-level definitions of binge-watching we find in the literature. Next, we outline how to introduce these features into Recommender Systems. We use two approaches: for models that do not use features by default, we extend the User Rating Matrix adding this information as an extra user and item; for Factorization Machines, we add features straightforwardly. We then train models weighting their features so that their importance can be automatically learned during the hyper-parameter optimization.

Our experiments show that Machine Learning models like SLIM and Factorization Machines leverage the information on binge-worthy series and improve their recommendation quality, both in terms of accuracy and beyond accuracy metrics. We also highlight that the models do not use binge-watchers' information as it does not lead to improvements in recommendation quality.

Finally, we outline possible use cases and research questions that can benefit from the extraction of binge-watchers and binge-worthy series information.

Sommario

Conseguentemente alla crescita dei servizi media Over-The-Top, il modo in cui i contenuti sono trasmessi al pubblico è cambiato drasticamente. In queste piattaforme, gli episodi delle stagioni delle serie-TV vengono rilasciati simultaneamente di frequente. Di conseguenza, anche il comportamento del pubblico si è profondamente evoluto: sempre più utenti di queste piattaforme guardano diversi episodi di fila in ogni sessione. Questo fenomeno è definito come binge-watching, mentre le serie-TV binge-worthy sono serie che il più delle volte vengono guardate attraverso binge-watching. L'obiettivo di questa tesi consiste nel fornire metodi per l'identificazione dei binge-watcher e delle serie binge-worthy, e nell'esplorare la rilevanza di queste informazioni all'interno di Sistemi di Raccomandazione.

Innanzitutto, identifichiamo le singole sessioni in cui un utente guarda una certa serie, ed estraiamo sia i binge-watcher che le serie binge-worthy, utilizzando definizioni di alto livello che troviamo nella bibliografia. Successivamente, illustriamo come introdurre queste feature all'interno dei Sistemi di Raccomandazione. Adottiamo due approcci differenti: per quanto riguarda i modelli che non prevedono l'utilizzo di features, estendiamo la User Rating Matrix aggiungendo queste informazioni come se fossero un utente e un oggetto extra; per le Factorization Machines, possiamo utilizzare direttamente le features. In prosieguo, addestriamo i modelli pesando le feature affinché la loro importanza venga automaticamente imparata durante l'ottimizzazione degli iper-parametri.

I nostri esperimenti mostrano che i modelli di Machine Learning come SLIM e le Factorization Machines sfruttano le informazioni sulle serie binge-worthy e migliorano la qualità delle loro raccomandazioni, sia per quanto riguarda le metriche di accuracy e beyond-accuracy. Inoltre, risulta che i modelli non usino le informazioni sui binge-watchers, siccome non migliorano la qualità delle raccomandazioni.

Infine, illustriamo possibili casi d'uso e sviluppi di ricerca che possono trarre beneficio dall'estrazione di binge-watchers e serie binge-worthy.

Acknowledgements

First of all, I would like to thank Mario and Fernando, my supervisors, for all the support they have given me during this work. Working with you has been a wholesome experience, challenging but rewarding, which helped me grow professionally and personally.

I would like to thank Professor Cremonesi and the whole ContentWise Research team for the opportunity they have given to me, your patience, and your precious insights: I will keep treasure of them forever.

To Elisa, for always being by my side, cheering me up in the most challenging moments, and taking care of me like no one ever did. You gave me the strength to keep pushing forward no matter what, and I could never have done this without you.

To my family, for supporting me through all these years and giving me the possibility to study at Politecnico. You have always believed in me through my successes and failures.

To Corda, Tom, and Alle, my brothers, and band-mates of Mother Augusta, and to our manager Sad, for sharing with me some of the most awarding and beautiful experiences of my life.

To all of my friends, especially Fede, Umbi, Berto, Guido, Richi, Bea, Marta, Fra, Rossy, Manu, and Giorgia, for their sincerity, loyalty, and for always being there for me, through good and bad times.

To my "Brothers of Metal" Gimmy, Gianca, Kara, Rampi, Saylor, Dest, for sharing our passion for music and the many marvelous times together.

To my "amici del mare" Alfre, Angi, Dido and Gio, for the unique and lifelong bond I have with you, that goes through distances.

To Music, my deepest passion, my purest love, for giving me the energy, the drive, the motivation to push through, and for being my constant study companion.

To all the amazing and smart friends I met at Politecnico, sharing these years of joy and pain with you has been a ride I will never forget.

Contents

Chapter 1

Introduction

With the exponential rise in popularity of Over-The-Top media services over the last 15 years, the way content is delivered and how it is consumed changed dramatically. Originally, TV-series were broadcasted periodically via cable, and the public needed to wait a certain amount of time for the next episode to be transmitted. Nowadays, with platforms such as Netflix or Prime Video gaining a terrific amount of popularity and new ones released daily, every episode of TV-series' seasons is released simultaneously. In addition to this, people can stream shows wherever they are through these platforms and are not limited anymore to do it at home on TV. Consequently, TV-series have become a worldwide phenomenon. People have started to consume entire seasons as soon as they get released, watching several episodes in a short amount of time. This behavior is referred to as binge-watching [1].

One of these platforms' main strengths is suggesting content tailored to users' preferences through Recommender Systems. Hence, the goal of this thesis is to explore if determining which users are binge-watchers and which series are prone to be binge-watched is relevant to improve the quality of recommendations provided to users. We mainly focus on understanding how to extract such information and introducing it into state-of-the-art recommendation algorithms, checking how the performance metrics are impacted.

1.1 Context: Recommender Systems in an industrial scenario

This thesis was developed during an internship at ContentWise. ContentWise is a software company working in user experience personalization for video operators, digital publishers, and online retailers. It develops and

provides a UX Engine to Over-The-Top media services, tailoring the subscribers' experience to their tastes, both in terms of front-end and back-end with personalized recommendations.

Historically, research on binge-watching has been very active in Sociology and Psychology. It is not easy to find a definition that can help identify uniquely what a binge-watcher is. Up to our knowledge, binge-watching has not been explored nor exploited yet in the Recommender Systems literature. This work of thesis proposes a rigorous approach to identify binge-watching behavior and seeks to identify its impact on recommendation performance, in continuity to research in the Recommender Systems field, which aims to improve recommendations.

1.2 Scenario and problem statement

Having the highest amount of information possible on its users is crucial for every platform. It is relevant to know which content a user consumes and how it consumes it. As an example, Context-aware Recommender Systems rely on this assumption, and they demonstrate that how, when, and where the user interacts with the content improves the recommendation quality significantly[22]. However, the literature lacks methods to extract binge-watchers and binge-worthy series from a dataset. The definitions are highly general and complex to apply to a real-life dataset. The main problem is the absence of rigorous methods to identify the binge-watchers in a dataset.

The absence of well-defined approaches to identify such information in a dataset leads to a consequent non-existent exploration of its significance. At the moment, the research community does not provide any hints on whether this behavioral pattern improves recommendation quality or not. With the wide-spreading of binge-watching as an increasingly common behavior in popular culture, we believe it needs to be assessed from the perspective of evaluating its usefulness as side information of recommendation engines.

1.3 Contributions

We propose novel approaches to identify binge-watchers and binge-worthy series in our dataset with the knowledge acquired by analyzing the state-of-the-art definitions of binge-watchers and Recommender System algorithms. We convert the extracted information into features to be introduced into Recommender Systems. The extraction of binge-watching features in an industrial dataset has not received attention yet from the research commu-

nity, as well as the exploration and exploitation of them into Recommender Systems. Therefore, we hope to help the research and whoever may find it helpful to gather and use such information in their systems with this contribution.

First, we provide methods to prepare the data before the binge-watching features extraction, and we test the dataset correctness. For this purpose, we consider only serial content that has been watched up to a certain percentage. Next, we identify the watching session of each user for each series. Watching sessions are the sequence of episodes of the same series a user watches in a single sitting. Watching sessions are the preliminary entity over which binge-watchers and binge-worthy series are defined, based on the number of episodes watched in a single session. Therefore, we translate the binge-watcher definitions in the literature to algorithmic procedures, which we test on the extracted watching sessions. Finally, we identify the binge-watchers and binge-worthy series and extract features that models will use.

Afterward, we describe the different models we have built using binge-watching and binge-worthy features. We introduce a way to extend the User Raring Matrix to inject such information into models. We provide a detailed description of User Collaborative Filtering, Item Collaborative Filtering, and a particular implementation of a SLIM ElasticNet model with contextual information using the extended URM. Moreover, we explain how we introduced these features into LightFM, an implementation of a Factorization Machine model.

Finally, we carry out a performance study by training models with binge-watching features alone, with binge-worthy features alone, and with a combination of both, comparing the recommendation quality to the one provided by baselines.

1.4 Definitions

- **Over-The-Top media service**: media service offered directly to the viewers via the internet, without utilizing the normal broadcasting mechanisms.
- **User**: the individual that interacts with the Over-The-Top media service and watches content.
- **Binge-watching**: the act of watching serial content consuming multiple episodes in a single sitting.
- **Binge-watcher**: a user who binge-watches regularly.

- **Binge-worthy series**: a TV-series that is prone to be binge-watched frequently by users.

1.5 Thesis structure

The rest of the thesis is structured as follows:

- Chapter 2 introduces the state-of-the-art techniques from which this thesis starts. It presents them in-depth to provide the reader with the essential knowledge to fully understand the topics presented.
- Chapter 3 presents a description and an analysis of ContentWise Impressions dataset, together with the filtering strategies adopted to extract binge-watching information.
- Chapter 4 delineates the path from watching sessions extraction through binge-watcher and binge-worthy series identification to the crafting of features that will be used in models. It highlights statistics over the features built as well.
- Chapter 5 outlines the adopted approach to introduce binge-watching features into models, reporting their architecture and decisions that lead us to implement our solution.
- Chapter 6 marks out the experimental setup, hence how the solution is implemented and evaluated.
- Chapter 7 illustrates the obtained results and compares them to the ones provided by baseline models.
- Chapter 8 sums up our work's contributions and limitations and suggests future research directions.

Chapter 2

State of the Art

2.1 Recommender Systems

Recommender Systems are a set of information filtering systems and techniques that aim to provide suggestions for items to be of use for a user [18]. These suggestions relate to various decision-making processes, like what music to listen to, what video to watch, or what item to purchase, for instance. Recommender Systems are widely used in domains where content personalization for each user is crucial, such as e-commerce websites, Over-The-Top media services, social networks, or online advertising.

Recommender Systems use different models and approaches, but the problem they solve is always the same: recommend items to users. Among the different sub-classes of Recommender Systems, we identify and describe one of the main ones: Collaborative Filtering [18].

2.1.1 Explicit and implicit feedback

To effectively develop a RS, user preferences need to be learned directly by users evaluating the items or without any user involvement [7].

Explicit feedback requires the users to explicitly evaluate items [7], while implicit feedback is derived from monitoring and analyzing users' activities [7]. On one hand, implicit feedback is easier to gather since it does not require direct user involvement. It is generally much more abundant than explicit feedback for the same reason, so it has become standard nowadays, though explicit feedback is still present in many systems [7]. On the other hand, explicit feedback allows the presence of *negative feedback*, since users can express their preferences over items. In contrast, in implicit feedback, users cannot express their preferences since only positive interactions are present [7].

2.1.2 User Rating Matrix

The User Rating Matrix (URM) encodes users' preference for items in the catalog of a RS, with shape $(number\ of\ users, number\ of\ items)$. It can be explicit if the feedback it contains is explicit or implicit if it contains implicit feedback.

URMs are usually the input of a recommendation algorithm. A RS can learn the users' preferences from the URM and predict the preference of the items each user has not yet interacted with.

2.1.3 Collaborative Filtering

Collaborative Filtering is a class of Recommender Systems that leverages the items preferred by other users as data to predict users' preferences. The core idea of Collaborative Filtering is to recommend to a user an item that other users with similar tastes have liked in the past [18]. If two users u and v both liked items i and j, the preferences of u and v are similar. If two users u and v have similar preferences, and user u has liked item i, it is likely that also user v will like that item.

There are two main sub-types of Collaborative Filtering Recommender Systems, memory-based, and model-based approaches:

- Memory-based approaches use user's rating data to compute the similarity between users or items and predict the user preference accordingly. We can further divide them into User-based or Item-based [18].
- Model-based approaches use models developed with different algorithms to predict users' rating on unrated items [18].

Memory-based approaches use a similarity measure to create the similarity matrix between users or items. The most common ones are Cosine similarity [18], Jaccard similarity [6], Dice similarity [2], Asymmetric similarity [4] and Tversky similarity [21].

User-based Collaborative Filtering

User-based Collaborative Filtering is a memory based approach that predicts the rating $r_{u,i}$ of a user u for a new item i using the ratings given to i by user that are most similar to u, called nearest-neighbors. Supposing we have computed a similarity measure between each user, $w_{u,v}$ represents the similarity between user u and user v, given that $u \neq v$, the k-nearest-neighbors (KNN) of u, denoted by $\mathcal{N}(u)$, are the k users v that have the highest similarity $w_{u,v}$ to u. The ratings $r_{u,i}$ using only the k most similar

users that have rated the item i, $\mathcal{N}_i(u)$. The rating $r_{u,i}$ are estimated as the average rating given to i by these neighbors: [18]

$$\hat{r}_{u,i} = \frac{1}{|\mathcal{N}_i(u)|} \sum_{v \in \mathcal{N}_i(u)} w_{u,v} r_{vi} \tag{2.1}$$

Usually, these ratings are normalized:

$$\hat{r}_{u,i} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{u,v} r_{v,i}}{\sum_{v \in \mathcal{N}_i(u)} |w_{u,v}|} \tag{2.2}$$

Item-based Collaborative Filtering

Item-based Collaborative Filtering method is the dual approach of the User-based one, but instead of relying on the opinion of likely-minded users, they look at ratings given to similar items. It is a memory-based collaborative filtering approach. Again, some similarity measure needs to be computed, but instead of computing it between user, $w_{i,j}$ will be the similarity between item i and item j (such that $i \neq j$). $\mathcal{N}_u(i)$ denotes the items rated by user u most similar to item i. Again, the predicted rating of u for item i will be obtained as a weighted average of the ratings by u of items from $\mathcal{N}_u(i)$: [18]

$$\hat{r}_{u,i} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{i,j} r_{u,j}}{\sum_{j \in \mathcal{N}_u(j)} |w_{i,j}|} \tag{2.3}$$

Matrix Factorization

Matrix Factorization (MF) is a model-based approach that aims to characterize both items and users with latent factors inferred from the dataset itself. Considering a latent factor space of dimensionality f, MF maps all user and items to vectors $v \in \mathbb{R}^f$. Then, given N users and M items, we consider $U \in \mathbb{R}_{N,f}$ and $V \in \mathbb{R}_{M,f}$, two matrices respectively for users and items [8].

We call each row vector v_i for these matrices a *latent vector*, where i is either a user or an item. Instead, columns of these matrices correspond to *latent factors*. The main idea is to predict the rating of user i for item j as a scalar product between user latent vector v_i and an item latent vector v_j. By expanding this idea, as shown in Figure 2.1, we obtain each user has predicted ratings for each item.

$$\hat{R} \approx UV^T \tag{2.4}$$

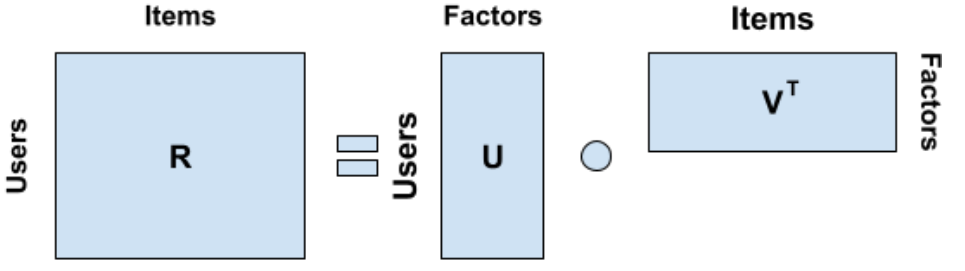


Figure 2.1: Rating prediction in MF. The predicted ratings (R) are equal to the product of the matrix with user latent factors (U) and item latent factors (V)

The main goal is to have matrix $\hat{R}$ as an approximation of the User Rating Matrix, the sparse matrix $R \in \mathbb{R}_{M,N}$ whose elements are the ratings $r_{i,j}$ of user i for item j.

2.1.4 SLIM

Sparse Linear Method (SLIM) is an algorithm to recommend the Top-N items. Even though it builds a model, it mimics the functioning of a memory-based Collaborative Filtering technique like the one described in Section 2.1.3, creating a model of the item-item similarity matrix learnt through the following optimization criteria:

$$\min_{W} \quad \frac{1}{2} \|URM - URM \cdot W\|_F^2 + \lambda \|w\|_1 \tag{2.5a}$$

$$\text{subject to} \quad W \geq 0, \tag{2.5b}$$

$$diag(W) = 0 \tag{2.5c}$$

where $\|\cdot\|_F$ is the Frobenius matrix Frobenius norm, which acts as a regularizer on the matrix, W is the item-item similarity matrix, $\|W\|_1$ is the entry-wise l_1-norm of W, and URM is the User Rating Matrix. The optimization is subject to 2.5b because the similarity between two items cannot be negative and to 2.5c because it is preferable to avoid a trivial solution where the similarity matrix is an identity matrix [13].

Once the optimization problem is solved and W is found, the predicted rating matrix is computed accordingly to the following equation [13]:

$$\hat{R} = URM \cdot W \tag{2.6}$$

Finally, according to the predicted ratings, we can order in decreasing order the Top-N items with which the user has not yet interacted and recommend them.

2.2 Context-aware Recommender Systems

Context-aware Recommender Systems (CARS) are an extension of Recommender Systems that considers users' specific situations to tailor recommendations accordingly and improve their quality [22]. CARS assumes that a user's item rating is not just a function of the user and the item but also of the context under which it was rated [22]. A user's preference on a given item might strongly rely on the context and vary from one to another [22]. For example, a user might give a different rating to a particular movie whether it watches it alone or with friends, during the day or after dinner.

2.2.1 Factorization Machines

Factorization Machines are a general recommendation algorithm working with any real feature vector, decomposing the rating matrix as a product of user and item vectors containing features [17]. Like CARS, they introduce contextual information into Recommender Systems [17]. They are similar to traditional MF models, but they can estimate predictions in problems with a high degree of sparsity and include other types of information, such as time of day, region, and others. Moreover, they can do so in linear time [17].

2.3 Recommender Systems Evaluation

Like any other Machine Learning sub-field, Recommender Systems require some specific evaluation technique to understand the recommendations' quality, and that is not a trivial task [5]. There are several metrics to evaluate a RS. We define some commonly used metrics.

Precision

Initially, to define precision it is necessary to recall the well known confusion matrix to evaluate classification problems [18], that also applies to Recommender Systems. The confusion matrix is defined in Table 2.1:

	Recommended	Not Recommended
Relevant	True Positive (TP)	False Negative (FN)
Irrelevant	False Positive (FP)	True Negative (TN)

Table 2.1: Confusion matrix for Recommender Systems. It shows all the possible outcomes of relevant and irrelevant items being recommended or not.

- **True Positive (TP)**: the algorithm recommended a relevant item;
- **True Negative (TN)**: the algorithm did not recommended an irrelevant item;
- **False Positive (FP)**: the algorithm recommended an irrelevant item;
- **False Negative (FN)**: the algorithm did not recommended a relevant item.

Subsequently, Precision is defined as the number of relevant recommended items divided by the total number of recommended items [5]:

$$Precision = \frac{Relevant\ recommended\ items}{Recommended\ items} = \frac{TP}{TP + FP}$$

Precision is usually evaluated by computing the metric just on the first K recommended items, *Precision@K*. Precision has values ranging from 0 (when no recommended item is relevant) to 1 (when every recommended item is relevant).

Recall

Recall is the ratio between relevant recommended items and the total relevant items. It is based on the definitions presented in Section 2.3.

$$Recall = \frac{Relevant\ recommended\ items}{Relevant\ items} = \frac{TP}{TP + FN}$$

Like Precision, Recall is generally used considering the first K recommended items, i.e., *Recall@K*. Its values range from 0 (no relevant item is recommended) to 1 (all relevant items are recommended, there are no False Negative items).

Mean Average Precision

Mean Average Precision (MAP) is one of the most relevant and widely used metrics for Recommender Systems [5]. First, let's define *AveragePrecision* at n ($AP@n$):

$$AP@n = \frac{\sum_{k=1}^{n} Precision@k \cdot Rel(k)}{Relevant\ items}$$

where $Rel(k) = 1$ if the item ranked at k is relevant, 0 otherwise. Subsequently, we can finally define $MAP@K$, similarly to Precision and Recall:

$$MAP@K = \frac{\sum_{i=1}^{N} AP_i@K}{N}$$

where N is the number of recommended lists of items. Values of MAP still range range between 0 and 1.

Mean Reciprocal Rank

Mean Reciprocal Rank (MRR) is an evaluation metric that produces a list of possible responses to a sample of queries, ordered by the probability of correctness. Like MAP, it gives more importance to items ranked higher than the ones ranked lower, specifically considering only the highest-ranked relevant item for each list of recommended items. Defining N as the number of recommendation lists a $rank_i$ as the rank of the highest-ranked relevant item in recommendation list i, MRR is defined as follows:

$$MRR = \frac{1}{|N|} \sum_{i=1}^{|N|} \frac{1}{rank_i}$$

To conclude, $MRR@K$ is defined as the Mean Reciprocal Rank computed only on the first K recommended items per each list. MRR again is defined in the range between 0 and 1.

F1

Precision and Recall can be simplified and merged into a single metric, called F1, that weights each one equally, creating a unified metric [5]:

$$F1 = \frac{2Precision \cdot Recall}{Precision + Recall}$$

Even though F1 makes the comparison between algorithms easier, it does not facilitate scenarios where Precision is more important than Recall or vice-versa. F1 ranges between 0 and 1.

Novelty

Novelty measures how "novel" a recommendation is, in terms of how popular the recommended item was in the train set. The Novelty of an item i, $Novelty_i$, is defined by the following equation:

$$Novelty_i = -\frac{\log_2(\frac{Popularity_i}{|Train|})}{|Items|}$$

where $Popularity_i$ is the popularity in the train set of item i, $|Train|$ is the number of interactions in the train set, and $|Items|$ is the number of items in the train set. To obtain the total Novelty, we sum each $Novelty_i$ for every recommended item and we average over the number of users:

$$Novelty = \frac{\sum_i Novelty_i}{N}$$

where N is the number of users. The Novelty ranges between 0 and 1.

A random recommendation will have the highest Novelty among every other recommendation since most of the items it would recommend are not the most popular ones on average: however, to build a good Recommender System, we need to strike a balance between familiar, popular items and discovery of new items the user has never heard of before [18]. The familiar items establish trust with the user, and the new ones allow the user to discover entirely new things that they might love [18].

2.4 Binge-watching

Generally speaking, binge-watching can be defined as the act of watching content, usually episodes of the same TV-series, for an extended period [20]. Up to now, the literature has not yet converged on a single rigorous definition to identify what is binge-watching [1]: some definitions take into account the number of episodes watched in a single session, like TiVo, Netflix, or Trouleau [20], others take into account the length of each of these sessions [16]. There may be a reason to part for both. For example, watching two episodes of an hour-long series would be classified in the same way as watching six episodes of a 20-minutes-long series [16]. Still, only the second would be considered a binge-watching behavior, to most people's view. There have been attempts to consider both the number of episodes watched and the watching session's length, which is arguably the most precise definition. Still, this has not been adopted by researchers [1].

The definitions in the literature are summed up in Table 2.2.

2.5 Correlation measures

2.5.1 Pearson-r correlation coefficient

Pearson-r correlation coefficient is a measure of linear correlation in two sets of data [14]. It is the covariance of two variables, divided by the product of their standard deviations; thus it is essentially a normalised measurement of the covariance, such that the result always has a value between -1 and 1 [14]. Pearson-r coefficient is defined as shown in Equation 2.7:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.7}$$

Research	**Binge-watching definition**	**Statistics**
Netflix	Watching 2-3 episodes of a tv series in a single session every few weeks	61% of users are regularly binge-watchers
TiVo	Watching 3 or more episodes in one day	91% of users report binge-watching as a common behavior, of whom 40% and 69% reported having at least one binge-watching session within a week and within a month respectively
Trouleau et al.	Normal behavior: 1-2 episode in a single sitting. Binge-watching: 3-7 episodes per sitting. Hyper binge-watching: ≥ 8 episodes per sitting	64% of users binge-watched at least once. 11% of users hyper binge-watched at least once. 7.6% of users all sessions are binge-watching ones. 20% of users binge-watch more than half of their sessions

Table 2.2: Statistics from different surveys. For each research, we show the corresponding binge-watching definition, and the statistics that presented in the paper/survey

2.5.2 Spearman-r correlation coefficient

Spearman-r correlation coefficient measures the rank correlation between the rankings of two variables [11]. It is strongly related to Pearson-r, since it can be defined as the Pearson correlation coefficient between two ranked variables:

$$r_s = \rho_{\mathrm{rg}_X, \mathrm{rg}_Y} = \frac{\mathrm{cov}(\mathrm{rg}_X, \mathrm{rg}_Y)}{\sigma_{\mathrm{rg}_X} \sigma_{\mathrm{rg}_Y}} \tag{2.8}$$

where X and Y are the unranked variables, rg_X, rg_Y are respectively the ranked variables of X and Y by appyling ranking operator rg_V with $V = X$ and $V = Y$, r_s is the Spearman-r correlation coefficient, and it is equal to $\rho_{\mathrm{rg}_X, \mathrm{rg}_Y}$, the Pearson correlation coefficient applied to the ranked variables rg_X, rg_Y. Overall, it is the covariance of the ranked variables rg_X and rg_Y divided by the product between the variance of each of the two ranked variables.

2.5.3 Kendall-τ correlation coefficient

Kendall-τ correlation coefficient is a statistic used to measure the quota of couples that are in the same relative positions in both rankings, i.e., the ordinal association between two measured quantities [9].

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{\binom{n}{2}} \tag{2.9}$$

where $(x_1, y_1), ..., (x_n, y_n)$ is a set of observations of the joint variables X and Y (ties are eliminated by considering the unique values of x_i and y_i for simplicity). Any pair of observations (x_i, y_i) and (x_j, y_j), where $i < j$, are said to be concordant if either both $x_i > x_j$ and $y_i > y_j$ holds or both $x_i < x_j$ and $y_i < y_j$ holds, otherwise are said to be discordant.

Tau-B

Tau-B is a variant pf Kendall-τ that makes adjustments for ties:

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}} \tag{2.10}$$

where

$n_0 = n(n-1)/2$

$n_1 = \sum_i t_i(t_i - 1)/2$

$n_2 = \sum_j u_j(u_j - 1)/2$

$n_c =$ Number of concordant pairs

$n_d =$ Number of discordant pairs

$t_i =$ Number of tied values in the i^{th} group of ties for the first quantity

$u_j =$ Number of tied values in the j^{th} group of ties for the second quantity

Chapter 3

Data and and preliminaries to feature crafting

3.1 Dataset

In this thesis, we base our analysis and tests over the *ContentWise Impressions* dataset, an industrial dataset collected from an OTT media service [15]. This dataset's peculiarity is that it is publicly available on GitHub[1]. The data was collected throughout four months [15] and will be described in detail throughout this section.

3.1.1 Data description

ContentWise Impressions is made of two separate datasets. For this thesis, we used the interactions dataset, which is structured as follows [15]:

- **Users**: users are the registered accounts for the OTT. Each can share its account with friends and family and use it on various devices. An anonymous numerical identifier represents every user;
- **Items**: items represent the media content provided by the service to the users of its platform. An anonymized numerical identifier identifies each item. Items vary in four different types:
  - *Movies*;
  - *Movies and clips*;
  - *TV movies or shows*;
  - *Episodes of TV series.*

[1] https://github.com/ContentWise/contentwise-impressions

An integer identifier identifies these categories with values 0, 1, 2, 3, respectively.

- **Interactions with the item**: User actions over the OTT. They are associated with a *timestamp*, i.e., the date and time when each interaction occurred. Interactions can be of four different types:

  - *View*: describe the fact that a user has watched a certain item. They are associated with a *vision factor*, that describes the point in which the user stopped viewing the item as a real number between 0 and 1. If a user stops watching near the end, the vision factor will be close to 1; instead, if a user stops watching right after the beginning, it will be close to 0.;
  - *Access*: the user accessed the item details;
  - *Purchase*: a user has purchased the item in the catalog of the relative interaction. In fact, some items need to be purchased;
  - *Rating*: the item has been rated explicitly on a scale from 1 to 5 by the user.

  These four categories are respectively identified with values 0, 1, 2, and 3.

Now we present in Tables 3.1, 3.2 and 3.3, meaningful statistics over the unfiltered dataset. Finally, in Table 3.4 [15] we describe how the dataset is digitally represented. These tables are extracted from the dataset's article [15].

Interaction Type	Count	Percentage
View	6,122,105	58.54%
Access	4,105,530	39.26%
Purchase	221,066	2.11%
Rating	9,109	0.09%
Total	10,457,810	100%

Table 3.1: Number of interactions grouped by their type.

3.2 Filtering

As stated in Section 2.4, the concept of binge-watchers is defined over users that consume serial content. Hence, we must consider only relevant inter-

Item Type	Count	Percentage
Episodes of TV series	9,076,428	86.79%
Movies	987,518	9.44%
TV Movies and shows	162,574	1.56%
Movies and clips in series	231,290	2.21%
Total	10,457,810	100%

Table 3.2: Number of interactions grouped by the item type.

Item Type	Count	Percentage
Episodes of TV series	123,831	85.36%
Movies	13,733	9.47%
TV Movies and shows	5,722	3.94%
Movies and clips in series	1,788	1.23%
Total	145,074	100%

Table 3.3: Number of items grouped by their type.

actions from which we can extract binge-watchers, discarding movies, or purchase interactions, for instance.

Overall, the preliminary step to extract binge-watchers is to filter the dataset accordingly to the definition of binge-watching. The relevant interactions to keep for this type of serial bounded analysis are those with the following characteristics:

- **TV series**: binge-watching is generally defined [20] as watching multiple episodes of a TV series in a short amount of time; hence for this analysis, we should only keep the interactions that are episodes of a TV series;

- **series length $\geq$ 3**: we set a threshold of at least three episodes since there might be some outliers in the dataset. For example, there are items labeled as TV series but with just 1 or 2 episodes: those are most likely movies misclassified as TV series;

- **view type**: we keep view interactions since we are interested in the content viewed by a user, not on the content purchased, rated, or for which they accessed the details;

- **vision factor $\geq$ 0.9**: to identify a binge-watching session, we only consider interactions where the user has viewed the episode completely.

Column name	Description
utc_ts_milliseconds	UTC timestamp in milliseconds. Index of the dataset
user_id	Numerical identifier of users
item_id	Numerical identifier of items
series_id	Numerical identifier of series
recommendation_id	Numerical identifier of the impression. Set to -1 for rows where the impression is not present
episode_number	Episode number of the item
item_type	Number ranging from 0 to 3, describes the category the item belongs to
interaction_type	Number ranging from 0 to 3, expresses the category of the interaction
explicit_rating	Rating from 1 to 5, with 0.5 step. Set to -1 for interactions without an explicit rating

Table 3.4: How the dataset is digitally represented. We show its column names and a qualitative description of what each column contains.

A vision factor of 0.9 means that the user stopped watching the episode at 90% of its length. We consider that there are credits and sometimes anticipations on the next episode at the end of each episode. It is a percentage decided coherently to previous studies on the dataset[19]. For example, suppose a user watched the first two minutes of every episode of a series in 2 hours without watching any episode completely. In that case, we should not consider this as a binge-watching session; neither could we conclude if the user is a binge-watcher or not based on these interactions.

In Table 3.5, we show the number of interactions, users, and series remaining from the original unfiltered dataset, both in terms of absolute value and percentage for the unfiltered dataset, when we apply the filtering described in the first column of the table mentioned above.

3.3 Sequence reconstruction

We aim to test if episode numbers in the dataset are reliable through sequence reconstruction, for the algorithm we designed to extract watching sessions relies strongly on the assumption that these numbers are correct. Furthermore, we would like to understand more in-depth the user behavior,

	Interactions	Users	Series
Original dataset	10.457.810 (100%)	42.153 (100%)	28.877 (100%)
TV series dataset	9.076.428 (86.79%)	39.430 (93.54%)	9.093 (31.49%)
TV series, view interactions, vision factor $\neq$ -1, series length $\geq$ 3	5.337.395 (58.81%)	38.327 (90.92%)	6.741 (23.34%)
TV series, view interactions, vision factor $\geq$ 0.9, series length $\geq$ 3	4.040.335 (38.63%)	36.988 (87.75%)	6.399 (22.16%)

Table 3.5: On the first column, highlighted in bold, we show the filtering we are applying. The table entries show the absolute number of interactions, users, and series and the percentage with respect to the original dataset.

and with sequence reconstruction, we can extract a metric of how likely a user watches a TV series from start to finish, watching all episodes sequentially, without skipping episodes and without doing any re-watches. We refer to this watching pattern as a *perfectly sequential sequence (PSS)*.

3.3.1 Pre-processing: readjusting the episode numbers

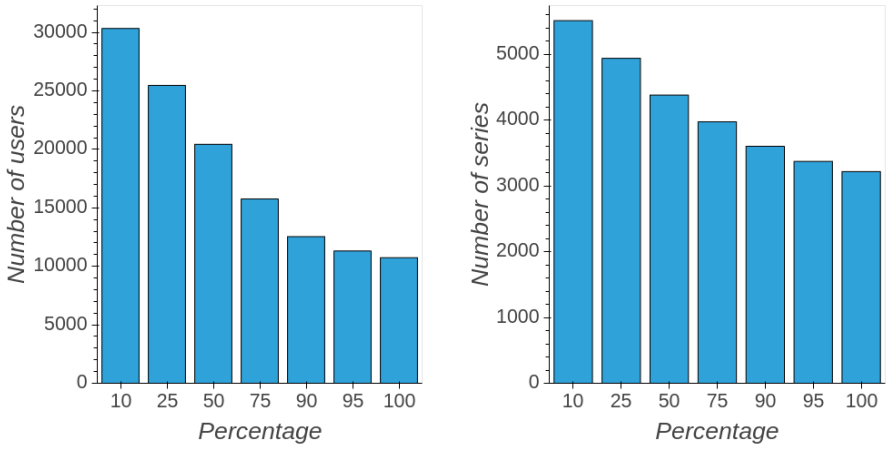
Our analysis on the dataset found that no series had an episode number equal to 1; every series had as a minimum episode number 2. Consequently, the series length of each series did not reflect the actual number of episodes of that series correctly, being it equal to the maximum episode number. The cause of this is not described in the original paper of the dataset [15]. We believe that episode number 1 was used during data gathering as a fictitious episode that acted as a placeholder for the series object.

As a preliminary step for sequence reconstruction, which is shown in the following section, we moved every episode number back by one. We decreased the length of each series by one as well in the filtered dataset. We now have episode numbers ranging from 1 to the actual length of the series instead of ranging between 2 and the series length + 1.

3.3.2 Method

We considered all the interactions from the filtered dataset where users have watched a series entirely. For each user-series pair, every episode number

between 1 and series length needs to be present at least one time. To ease notation, we call this filtering over the dataset as the *full view dataset*. In Figures 3.1(a) and 3.1(b) we show the number of users and series that the dataset has if we consider interactions where users have watched a certain percentage of series. Approximately 11K users have watched some series entirely at least once (100% percentage), and 3.2K series have wholly been watched at least once by some user.



(a) Number of users that have watched at least % of any series

(b) Number of series watched at least % by any users

Figure 3.1: Bar plots that show how users watches series 3.1(a) and how series are watched by users 3.1(b)

We take users that have watched a series completely to test that the episode number is reliable, and then we can infer that holds up as well for all users and every interaction.

From a high-level perspective, the algorithm to test the correctness of episode numbers starting from the dataset where every interaction is from a user that has watched a series entirely works as follows:

1. For each series s:
   (a) Create a square ($series\ length$, $series\ length$) count matrix with row indices as the episode watched and column index as the position the episode was watched, and as an element the number of times the i-th episode has been watched as j-th episode;
   (b) For every user u that watched every episode of series s:
      i. Extract the user-series watching sequences for the pair (u, s);

ii. For each of that sequence, update the count matrix of series s.

(c) Normalize each row of the count matrix to obtain a probability matrix: it will contain the probability that the i-th episode will be watched as j-th episode;

(d) Create an identity matrix of size (*series length*, *series length*), representing the probability matrix that s would have if it were watched only in PSSs;

(e) Compute the Pearson-r correlation coefficient (presented in Section 2.5.1) that will represent of much the probability matrix is likely to be watched in a PSS.

We now present in detail the main steps, using an example to make things easier more understandable.

Extracting watching sequences

First, we need to group by users and series pairs the full view dataset and order each group by the interactions' timestamp.

A group of user-series interactions looks like the list of episodes the user has watched for that series, ordered temporally. For example, a group of user-series interactions for a series of length 16 is shown in Table 3.6.

1	2	3	4	6	7	8	9	10	11	12	13	14	4	5	9	10	11	12	13	14	15	16	13	14	15	16

Table 3.6: Example of a group of user-series interactions for a series of length 16.

Every time the user "goes back", hence watches a previous episode instead of a subsequent one, it means a new sequence has begun.

In Table 3.6, we identify three different sequences, between episodes 1 and 14, 4 and 16, and 13 and 16. In Table 3.7, the separation between different sequences is highlighted with a vertical border, instead when a user skips some episodes, i.e., when they watch instead of the immediate following one another subsequent episode, we highlight it in bold:

1	2	3	4	**6**	7	8	9	10	11	12	13	14	4	5	**9**	10	11	12	13	14	15	16	13	14	15	16

Table 3.7: Sequence extraction example. There are three different sequences, between 1 and 14, 4 and 16, 13 and 16. Values are highlighted in bold when the user skips an episode.

Count matrix

For each series, it is initialized a square count matrix of size *series length* and with all entries equal to zero. It contains as row indices the episode watched, as column indices the position in the corresponding sequence where the episode was watched, as elements the number of times the i-th episode has been watched as the j-th episode.

The update rule for the count matrix, for every sequence of a (user, series) pair, is the following:

for $sequences \in sequences$ **do**
  for $pos \in (0, |sequence|)$ **do**
    $countmatrix[sequence[pos], pos] \leftarrow [sequence[pos], pos] + 1$
  end for
end for

For the sequence considered in Table 3.7, the relative count matrix will be the one in Table 3.8.

	Watched as															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Episode number																
1	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
4	**1**	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
5	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
9	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	0
10	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0
11	0	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	0
12	0	0	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0
13	**1**	0	0	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0
14	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
15	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0
16	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0

Table 3.8: Count matrix of the example sequence [1 2 3 4 6 7 8 9 10 11 12 13 14 — 4 5 9 10 11 12 13 14 15 16 — 13 14 15 16]. From the Table we can see that both episode 1 and 4 has been watched once as the first episode in the sequence.

For example, following the count matrix update rule, the entry in position 1 of the second sequence is 4, so we will increase the count matrix in position (4, 1) by 1.

Probability matrix

The probability matrix is a normalization over the count matrix's rows: we divide each row by the sum of entry of the count matrix for that row. Each row then adds up to one.

In the end, the probability matrix for pair (i, j) is the probability of watching episode i as the j-th episode of a sequence.

In Table 3.9, we show the probability matrix for the example we are showing.

	Watched as															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Episode number																
1	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
4	**0.5**	0	0	**0.5**	0	0	0	0	0	0	0	0	0	0	0	0
5	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
9	0	0	**0.5**	0	0	0	0	**0.5**	0	0	0	0	0	0	0	0
10	0	0	0	**0.5**	0	0	0	0	**0.5**	0	0	0	0	0	0	0
11	0	0	0	0	**0.5**	0	0	0	0	**0.5**	0	0	0	0	0	0
12	0	0	0	0	0	**0.5**	0	0	0	0	**0.5**	0	0	0	0	0
13	**0.33**	0	0	0	0	0	**0.33**	0	0	0	0	**0.33**	0	0	0	0
14	0	**0.5**	0	0	0	0	0	**0.5**	0	0	0	0	0	0	0	0
15	0	0	**0.5**	0	0	0	0	0	**0.5**	0	0	0	0	0	0	0
16	0	0	0	**0.5**	0	0	0	0	0	**0.5**	0	0	0	0	0	0

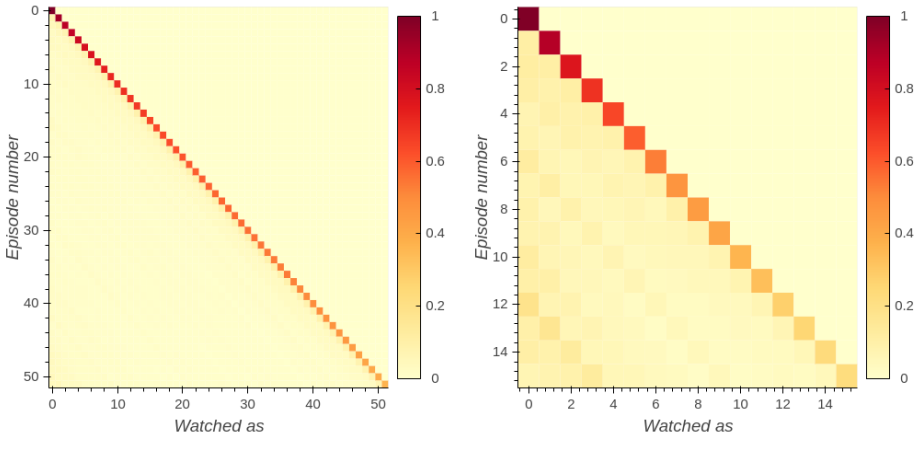
Table 3.9: Probability matrix of the example sequence [1 2 3 4 6 7 8 9 10 11 12 13 14 — 4 5 9 10 11 12 13 14 15 16 — 13 14 15 16]. From the Table we see that the episode 13 has a probability of 0.33 *to be either the first, seventh, or twelfth episode to be watched.*

Pearson-r correlation coefficient as a measure of seriality

We define *seriality* as the likelihood of a series to be watched in the order of its episode numbers. To have a measure of how a series is serial, we recurred to Pearson-r correlation coefficient, as defined in Section 2.5.1: by computing this coefficient between the probability matrix of a series and an identity matrix of the same shape, which represents a PSS, we obtain a measure representing if the series linearly correlates to one watched in a PSS.

3.3.3 Results

First, we plot different probability matrix heatmaps for series with a high amount of interactions. Here are two examples of how those heatmaps look like:



(a) Probability matrix - 117K interactions - Pearson-r = 0.961

(b) Probability matrix - 16K interactions - Pearson-r = 0.858

Figure 3.2: Heatmaps showing the values of two probability matrices. The x-axis represents the order in which the episode is watched in a sequence. The y-axis the episode number. Both matrices are in logarithmic scale. Higher values in red tones and lower probabilities in yellow tones.

In Figure 3.2(a) the probability matrix resembles a diagonal matrix; consequently, it has a Pearson-r value close to 1. In Figure 3.2(b) instead, we notice there are some interactions below the main diagonal. However, the matrix density is still on the main diagonal, and we get a lower Pearson-r-value than the one in 3.2(a), though mostly it has been watched in a PSS fashion.

We compute the Pearson-r value for different chunks of the number of interactions inside the full view dataset. Then, the Pearson-r is weighted by the number of interactions to give more importance to series with many interactions: each series' Pearson-r is multiplied by the number of interactions that series has in the full view dataset. We sum all this so computed Pearson-r values. Finally, we divide this sum by the sum of all interactions in the full view dataset. We show the results obtained in Table 3.10:

To conclude, the results shown in Table 3.10 highlight that series tend to be watched from start to finish, following the episode sequence most of the time. However, not every TV-series is episodic, i.e., have a story that

	Weighted Pearson-r
All series in full view dataset	0.7533
Series with > 100 interactions	0.7581
Series with > 500 interactions	0.7653
Series with > 1000 interactions	0.7692
Series with > 10000 interactions	0.8306

Table 3.10: Weighted Pearson-r values of all series with at least a certain number of interactions.

develops through episodes and seasons. Still, there exists also anthology series that present different stories and different characters from episode to episode, and so the order in which those are watched is not relevant. The dataset does not include extra information about series, e.g., if they are episodic not, but we can assume that a series with a low Pearson-r is likely to be an anthology series.

A weighted Pearson-r value of 0.7553 for all series and 0.8306 for series with more than 10K interactions make us safely assume that the episode numbers are reliable throughout the dataset.

Chapter 4

Binge-watchers identification and feature crafting

In Section 3.3.2, we showed that the episode number in the dataset is reliable, based on the results shown in Section 3.3.3. These results are crucial for watching session extraction, which is presented through this chapter, as it strongly relies on the assumption that episode numbers are correct. Finally, we describe how to extract binge-watchers and binge-worthy series.

4.1 Watching sessions

Watching sessions describe the act of watching one or multiple episodes of the *same* TV-series in a single sitting. How many episodes are watched in a single watching session will be the main element to distinguish binge-watching sessions from normal ones.

4.1.1 Watching sessions extraction

In this thesis, watching sessions are extracted on user-series pairs. For each user-series, the watching sessions will be an array of arrays. Every internal array represents a single watching session, and it contains the sequence of episodes watched in the watching order.

To understand what separates different sessions, we need to define a *session threshold*, which expresses the hours that can pass between two following interactions to be considered part of the same session.

Choosing a session threshold of the right size is very important. For example, suppose that a user watches consecutively two episodes in the morning and two episodes in the evening. Realistically, we would identify

two watching sessions, each composed of two episodes. Considering a session threshold too small, shorter than the episode length, we would extract four sessions, each containing one episode. Considering a session threshold too large, of 24 hours, for instance, we would extract a single session with four episodes. Finally, considering a more balanced threshold, we would extract two sessions with two episodes that would correctly reflect the user's behavior.

We tested different session threshold values. Based on past experience, through internal studies at ContentWise, we decided to proceed with a value of 4 hours because it represents a well-balanced trade-off for the problems mentioned above.

Example

We have a sequence of the various temporally ordered interactions for a user with a series, with just the episode number and the relative timestamp for simplicity.

Episode number	**Timestamp**
1	01/01/2020 10:00
2	01/01/2020 12:00
3	01/01/2020 15:30
4	01/01/2020 20:00
5	02/01/2020 20:00

Table 4.1: Example of a sequence of user-series interactions ordered by their timestamp.

We set a session threshold of 4 hours. 2 hours pass between the interactions with episodes 1 and 2, which is smaller than the session threshold. Between the watching of episodes 2 and 3 pass 2.5 hours, so 1, 2, and 3 belong to the same watching session. Instead, the watching of episode 4 is 4.5 hours after the watching of episode 3, so it does not belong to the previous watching session. Also, the watching of episodes 4 and 5 belong to two different watching sessions.

In conclusion, the three sessions extracted in the example are the following:

- $[1, 2, 3]$
- $[4]$
- $[5]$

The resulting array containing watching sessions is $[[1, 2, 3], [4], [5]]$.

Episode number	Timestamp
1	01/01/2020 10:00
2	01/01/2020 12:00
3	01/01/2020 15:30
4	01/01/2020 20:00
5	02/01/2020 20:00

Table 4.2: Watching session extraction for the example presented. The horizontal lines in the following table separate the different watching sessions

4.2 Binge-watchers and binge-worthy series identification

The concept of binge-worthiness is not as vastly defined in literature as it is the one of binge-watching. Consequently, we define a binge-worthy series as a series prone to be binge-watched frequently by users.

In the following section, we define the user-series engagement table, a table containing, for each user-series pair present in the dataset, the corresponding watching sessions, alongside different statistics to have a complete picture of how much a user is involved when watching a particular series.

4.2.1 User-series engagement table

The last preliminary step to extract binge-watchers and binge-worthy series is building a support user-series engagement table, which will act as a supporting tool for building binge-watching features, as shown later in this chapter. We build this table considering only user-series pairs where users have watched at least 50% of episodes of the corresponding series: for each user u and series s, we have a list of interactions $i_{u,s}$, ordered by their timestamp. From $i_{u,s}$ we will compute every metric in the user-series engagement table for the entry u, s.

This table contains different metrics and insights measured from the dataset, grouped by categories. All the metrics shown are computed for each user-series pair.

Time interval metrics

These metrics keep track of the difference between timestamps inside the interactions for each user-series pair. These time intervals can be relative to interactions or episodes. We compute:

- the difference between the *first time* the user watched the *first episode* and the first time the user watched the last episode;
- the difference between the *first time* the user watched the *first episode* and the last time the user watched the last episode;
- the difference between the first and the last time a user watched any episode.

Time interval averages

For each time interval metric presented above, we compute the time that passes on average between one episode and the other.

Episodes and interaction metrics

We also include the following metrics on the episodes:

- the total number of interactions $n\ interactions$;
- the number of episodes watched in terms of unique episode numbers present in the interactions list $episodes\ watched$;
- the difference between the number of interactions and the unique episodes watched $n\ interactions - episodes\ watched$;
- the percentage of episodes re-watched with respect to the unique episodes watched $\frac{n\ interactions - episodes\ watched}{episodes\ watched}$.

View sequences insights

With the methods presented in 3.3, we included:

- the view sequences;
- the number of sequences;
- a list containing the first episode of each sequence;
- a list containing a categorical interpretation of the first episode of each sequence. The possible values are *first, middle, last*, where *first* means it is the sequence begins with episode number 1, *last* with the last episode and *middle* with any episode in between.

Watching sessions elements

The watching session elements in the user-series engagement table are the ones through which we will extract binge-watchers and binge-worthy series.

To extract watching sessions, we used a *session threshold* of 4 hours. It contains:

- the watching sessions array;
- the number of watching sessions *n sessions*, i.e., the length of watching sessions array;
- the length of each watching session, i.e., an array containing for each watching session its length;
- the average length of the watching sessions;
- the time length of each watching session computed as the time differential between first and last episode within that session;
- the average time length of all watching sessions;
- an array that tells if each session is a PSS (for example, a session with episodes [1, 2, 3] is a perfectly sequential sequence, in which episodes are watched respecting their ordering; instead, sessions [1, 3, 4], [1, 1, 2], [1, 4, 3] are not);
- the percentage of PSSs with respect to the number of all watching sessions *n sessions*.

We apply the definitions of binge-watching sessions presented in Table 2.2 to understand how many of all sessions are binge-watching for the definitions mentioned above. We include:

- the number of watching sessions with length 3 or more (which corresponds to TiVo's definition [3]);
- the number of watching sessions with length from 2 to 6 (which corresponds to Netflix's definition [12]);
- the number of watching sessions with length from 3 to 7 (which corresponds to Trouleau et al.'s definition [20]).

4.2.2 Binge-watchers and binge-worthy thresholds definition

We construct the user-series engagement table using user-series interactions for users who watched at least 50% of that series and 4 hours of session threshold. There, we express the number of watching sessions defined as binge-watching sessions according to the literature definitions [3, 12, 20]. To distinguish if a user can be considered a binge-watcher or not, and if a series can be considered a binge-worthy one, we set up two thresholds.

Binge-watcher threshold

The threshold is expressed in percentage. A user is a binge-watcher if the ratio between the sum of all the binge-watching sessions for a specific definition divided by the total number of sessions for the user is greater than the binge-watcher threshold.

$$bingewatcher_{u,d} = \begin{cases} \text{True, if } \frac{\sum_{s \in \text{series(u)}} n\_bw\_sessions_{u,d,s}}{\sum_{s \in \text{series(u)}} n\_sessions_{u,s}} \geq threshold \\ \text{False, otherwise} \end{cases}$$

where $threshold$ is the binge-watcher threshold, $bingewatcher_{u,d}$ says if user u is a binge-watcher with respect to definition d, $n\_sessions_{u,s}$ is the number of sessions of u for series s and $n\_bw\_sessions_{u,d,s}$ is the number of binge-watching sessions of u for series s and definition d.

Binge-worthy threshold

The threshold is expressed in percentage. A series is binge-worthy if the ratio between the sum of all the binge-watching sessions of every user for a specific definition divided by the total number of sessions for that series is greater than the binge-worthy threshold.

$$bingeworthy_{s,d} = \begin{cases} \text{True, if } \frac{\sum_{u \in \text{users(s)}} n\_bw\_sessions_{u,d,s}}{\sum_{u \in \text{user(s)}} n\_sessions_{u,s}} \geq threshold \\ \text{False, otherwise} \end{cases}$$

where $threshold$ is the binge-worthy threshold, $bingeworthy_{s,d}$ says if series s is a binge-worthy series with respect to definition d, $n\_sessions_{u,s}$ is the number of sessions of u for series s and $n\_bw\_sessions_{u,d,s}$ is the number of binge-watching sessions of u for series s and definition d.

The two threshold definitions are dual since binge-worthiness itself is defined as a series that is binge-watched frequently.

4.2.3 Series grouping

The statistics on binge-worthy series are bonded to the series length. To present those statistics, we divide the series into groups based on the series length. We adopt two approaches to group the series:

- groups have the same number of interaction in the dataset;
- groups have the same number of series.

For each approach, we obtain three groups. In Table 4.3 we show how the groups are composed.

	Groups	Number of elements within interval
Same number of interactions	3 - 15	1,433,699
	16 - 38	1,366,412
	$\geq$ 39	1,266,914
Same number of series	3 - 11	2,220
	12 - 24	2,385
	$\geq$ 25	2,048

Table 4.3: Division of series into series length groups. There are three series groups: from 3 to 15, 16 to 38, and more than 39 episodes. In the table we show the number of interactions and series within each group.

4.2.4 Binge-watchers and binge-worthy series statistics

We obtain three different metrics to identify binge-watchers and binge-worthy series for each user-series pair, corresponding to each binge-watching definition. In this section, we present how we converge to a single one and use it to build features and the reasons beyond that choice. We base our decisions on the Tables 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10. The first being the binge-watchers statistics and the rest being the statistics of binge-worthiness of each series group by interactions and number of series, respectively.

We test different values for binge-watcher and binge-worthy thresholds presented in Section 4.2.2 and compare the percentage of binge-watchers with respect to the ones we find in the literature.

4.2.5 Final remarks on binge-watching definitions

In conclusion, based on the results in Table 4.4, we decide to adopt Trouleau's definition for binge-watchers. It was the only definition that included the percentage of users that binge-watched at a specific rate. We test and compare those results with the percentages obtained from our dataset: for Trouleau, 20% of users had at least half of their sessions as binge-watching

Binge-watcher threshold %	**Number of users**	**Binge-watchers≥ 3 TiVo**	**Binge-watchers 2 ÷ 6 Netflix**	**Binge-watchers 3 ÷ 7 Trouleau)**
10%	20432	18124 88.70%	18602 91.04%	16474 80.63%
30%	20432	13733 67.21%	15782 77.24%	9912 48.51%
50%	20432	9910 48.50%	9795 47.93%	4524 22.14%
70%	20432	5710 27.95%	3267 15.99%	1413 6.92%
95%	20432	2978 14.58%	1400 6.85%	954 4.67%
100%	20432	2887 14.12%	1399 6.84%	954 4.67%

Table 4.4: Binge-watcher statistics. Each row contains the threshold selected, the total number of users in the dataset, the absolute number of binge-watchers, and its percentage with respect to the total number of users, for each definition.

Binge-worthy threshold %	**Number of series**	**Binge-worthy ≥ 3 (TiVO)**	**Binge-worthy 2 ÷ 6 (Netflix)**	**Binge-worthy 3 ÷ 7 (Troueleau)**
10%	2142	2034 94.96%	1892 88.33%	1795 83.80%
30%	2142	1791 83.61%	1584 73.95%	1255 58.59%
50%	2142	1458 68.07%	1128 52.66%	647 30.21%
70%	2142	947 44.21%	441 20.59%	243 11.34%
100%	2142	444 20.73%	214 9.99%	159 7.42%

Table 4.5: Binge-worthy series statistics - Series length 3 ÷ 7 - Same number of interactions - 1433699 interactions. Each row contains the threshold selected, the total number of series in the dataset, the absolute number of binge-worthy series, and its percentage with respect to the total number of series, for each definition.

ones [20]. Data shown in Table 4.4 share the same results with Trouleau [20]. The same holds up as well for the 7.6% of users for whom all of their sessions are binge-watching ones: in our dataset, we reach 4.67%, which is close enough to assume that this definition is the best-suited one. Moreover, it is the most restrictive one since it provides an upper bound and a lower bound.

Consequently, we used Trouleau's definition to commute binge-watching and binge-worthiness information into features that will be plugged into models.

Binge-worthy threshold %	**Number of series**	**Binge-worthy ≥ 3 (TiVO)**	**Binge-worthy $2 \div 6$ (Netflix)**	**Binge-worthy $3 \div 7$ (Troueleau)**
10%	1903	1893 99.47%	1710 89.86%	1673 87.91%
30%	1903	1812 95.22%	1537 80.77%	1276 67.05%
50%	1903	1562 82.08%	987 51.87%	349 18.34%
70%	1903	836 43.93%	150 7.88%	44 2.31%
100%	1903	287 15.08%	14 0.74%	9 0.47%

Table 4.6: Binge-worthy series statistics - Series length $16 \div 38$ - Same number of interactions - 1366412 interactions. Each row contains the threshold selected, the total number of series in the dataset, the absolute number of binge-worthy series, and its percentage with respect to the total number of series, for each definition.

Binge-worthy threshold %	**Number of series**	**Binge-worthy ≥ 3 (TiVO)**	**Binge-worthy $2 \div 6$ (Netflix)**	**Binge-worthy $3 \div 7$ (Troueleau)**
10%	338	336 99.41%	318 94.08%	311 92.01%
30%	338	319 94.38%	269 79.59%	230 68.05%
50%	338	293 86.69%	128 37.87%	47 13.91%
70%	338	175 51.78%	11 3.25%	3 0.89%
100%	338	27 7.99%	2 0.59%	0 0%

Table 4.7: Binge-worthy series statistics - Series length ≥ 39 - Same number of interactions - 1266914 interactions. Each row contains the threshold selected, the total number of series in the dataset, the absolute number of binge-worthy series, and its percentage with respect to the total number of series, for each definition.

4.3 Binge-watching and binge-worthy features crafting

We build binge-watching features starting from the user-series engagement table. We now present two different versions for both binge-watching and binge-worthy features, one with boolean values 0 and 1, called the **implicit feature**, and the other with non-negative integers, called the **explicit feature**.

To compute those features, we proceeded in the following way:

- to build the **explicit binge-watching feature**, we count the number of binge-watching sessions for each user u among every series ;

Binge-worthy threshold %	Number of series	Binge-worthy ≥ 3 (TiVO)	Binge-worthy $2 \div 6$ (Netflix)	Binge-worthy $3 \div 7$ (Troueleau)
10%	1239	1147 92.57%	1118 90.23%	1051 84.83%
30%	1239	971 78.37%	969 78.21%	757 61.10%
50%	1239	764 61.66%	726 58.60%	429 34.62%
70%	1239	494 39.87%	344 27.76%	176 14.21%
100%	1239	262 21.15%	179 14.45%	123 9.93%

Table 4.8: Binge-worthy series statistics - Series length $3 \div 11$ - Same number of series - 2048 series in the unfiltered dataset - 468204 interactions. Each row contains the threshold selected, the total number of series in the dataset, the absolute number of binge-worthy series, and its percentage with respect to the total number of series, for each definition.

Binge-worthy threshold %	Number of series	Binge-worthy ≥ 3 (TiVO)	Binge-worthy $2 \div 6$ (Netflix)	Binge-worthy $3 \div 7$ (Troueleau)
10%	1987	1964 98.84%	1745 87.82%	1699 85.51%
30%	1987	1839 92.55%	1501 75.54%	1224 61.60%
50%	1987	1541 77.55%	1011 50.88%	432 21.74%
70%	1987	860 43.28%	198 9.96%	96 4.83%
100%	1987	340 17.11%	46 2.32%	42 2.11%

Table 4.9: Binge-worthy series statistics - Series length $12 \div 24$ - Same number of series - 2385 series in the unfiltered dataset - 1522736 interactions. Each row contains the threshold selected, the total number of series in the dataset, the absolute number of binge-worthy series, and its percentage with respect to the total number of series, for each definition.

- to build the **explicit binge-worthy feature**, we count the number of binge-watching sessions for each series s among every user;
- to build the **implicit binge-watching feature**, we divide the number of binge-watching sessions by the total number of sessions for each user u among every series. If this ratio, expressed in percentage, is greater than the 50% binge-watching threshold, u is a binge-watcher, and the value of the feature will be 1, otherwise 0;
- to build the **implicit binge-worthy feature**, we divide the number of binge-worthy sessions by the total number of sessions for each series

Binge-worthy threshold %	**Number of series**	**Binge-worthy ≥ 3 (TiVO)**	**Binge-worthy $2 \div 6$ (Netflix)**	**Binge-worthy $3 \div 7$ (Troueleau)**
10%	1157	1152 99.57%	1057 91.36%	1029 88.94%
30%	1157	1112 96.11%	920 79.52%	780 67.42%
50%	1157	1008 87.12%	506 43.73%	182 15.73%
70%	1157	6045 2.20%	60 5.19%	18 1.56%
100%	1157	156 13.48	5 0.43%	3 0.26%

Table 4.10: Binge-worthy series statistics - Series length ≥ 25 - Same number of series - 2048 series in the unfiltered dataset - 2049395 interactions. Each row contains the threshold selected, the total number of series in the dataset, the absolute number of binge-worthy series, and its percentage with respect to the total number of series, for each definition.

s among every user. If this ratio, expressed in percentage, is greater than the 50% binge-worthy threshold, s is a binge-worthy series, and the value of the feature will be 1, otherwise 0.

Finally, we will obtain two feature arrays, one with binge-watching features and the other with binge-worthy features, containing one entry for every user and one entry for every series.

The next chapter will explain how those features are plugged into models and how we evaluate their impact on recommendations.

Chapter 5

Approach

In Chapter 4 we described the road-map from the filtered dataset to feature extraction. In this chapter, instead, we describe the main approaches adopted to build models that can leverage those features. We present how we build models that can use binge-watching and binge-worthy features. We propose methods to fit binge-watching and binge-worthy information in the URM in Collaborative Filtering techniques. We then present a Context-aware SLIM model to include the features, and finally, an implementation of a Factorization Machine model.

5.1 Extension of the User Rating Matrix

As presented in Section 4.3, the binge-watching features are an array that contains the number of binge-watching sessions of a user for all series. The binge-watching features are an array containing the number of binge-worthy sessions of each series through all users.

We include this information in the recommendation algorithms by extending the URM as shown in Figure 5.1. We append to the URM with N users and M items the binge-watching array as a column array in position $M + 1$, as a fictitious item, and the binge-worthy array as a row array at position $N + 1$, as a fictitious user. In position $N + 1, N + 1$, we insert an extra 0 as a rating for the fictitious user and item, to avoid dimensional mismatches in the implementation. Consequently, we obtain an extended URM of size $N + 1, M + 1$.

We also extend the URM with one feature array at a time, as shown in Figure 5.2 and 5.3: the extended URM with binge-watching features has size $N, M + 1$; the extended URM with binge-watching features has size $N + 1, M$.

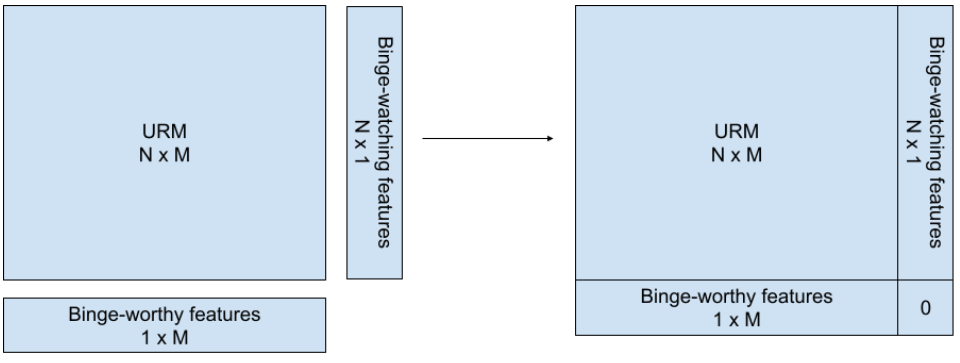


Figure 5.1: Including binge-watching and binge-worthy features in the URM

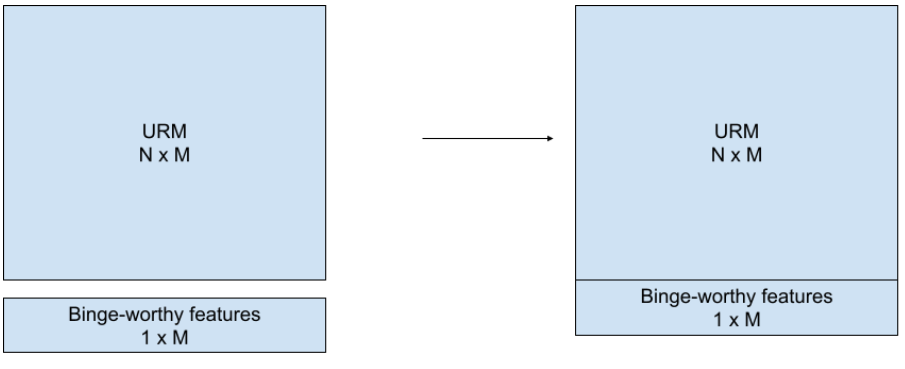


Figure 5.2: Including binge-watching features in the URM

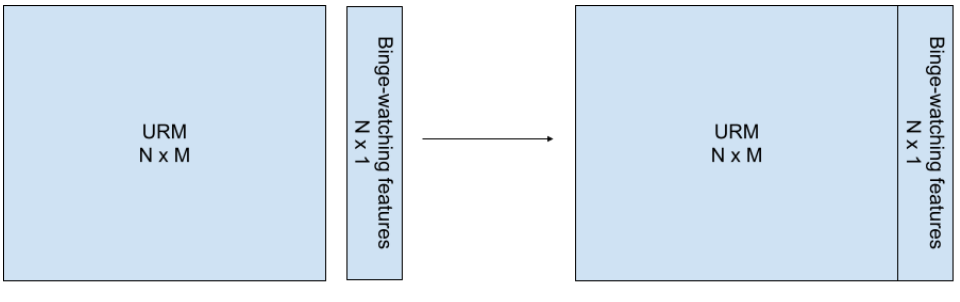


Figure 5.3: Including binge-worthy features in the URM

We implemented three version of models that use the extended URM: one using the binge-watching features, one using the binge-worthy features, one combining both.

5.2 Models using the extended URM

In this section, we outline how the models compute the predicted ratings using features, either by using the extended URM like SLIM, User and Item Collaborative Filtering, or by directly fitting the features into models that accept them, like Factorization Machines.

5.2.1 User Collaborative Filtering

User Collaborative Filtering with binge-watching and binge-worthy features

The extended URM with binge-watching and binge-worthy features is plugged into a User Collaborative Filtering model. It computes the user similarity as described in 2.1.3 using the extended URM instead of the normal URM: this leads to a similarity matrix of size $N+1, N+1$, where the extra user is the binge-worthy array. To compute the predicted rating matrix, the model computes the dot product between the user similarity and the extended URM. The predicted rating matrix has the same size as the extended URM, $N+1, M+1$.

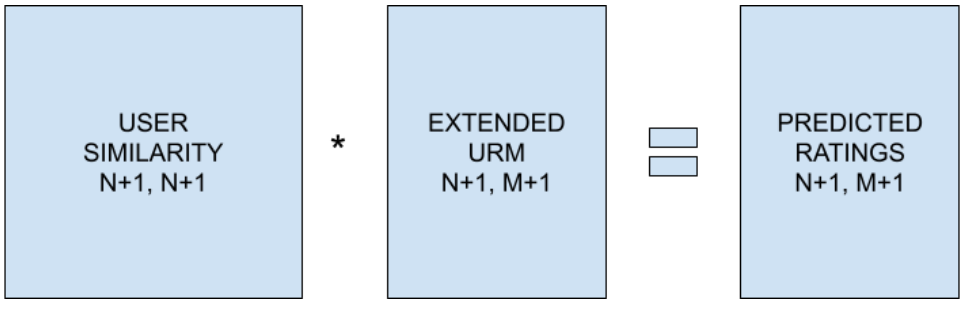


Figure 5.4: A visual representation of how the predicted ratings are calculated for User CF. Each square represents a matrix. The predicted ratings are the dot product of the similarity and the extended URM with binge-watching and binge-worthy features.

Since we do not want the last item to be recommended, which corresponds to the fictitious item representing the binge-watching feature array, we remove the $M+1$ column from the predicted ratings.

Finally, for each user u of the predicted rating matrix, we sort the items in a decreasing order accordingly to the ratings. We then recommend the top-k ranked items.

The fictitious $M+1$ user' scores corresponding to the binge-worthy features will not be relevant, as shown in Chapter 6 when we will be describing the evaluation methods.

User Collaborative Filtering with binge-watching features

Using only the binge-watching features means the extended URM has shape $N, M+1$. So the similarity will be the one of a regular User Collaborative Filtering recommender, with size N, N. The predicted rating matrix has size $N, M+1$ then.

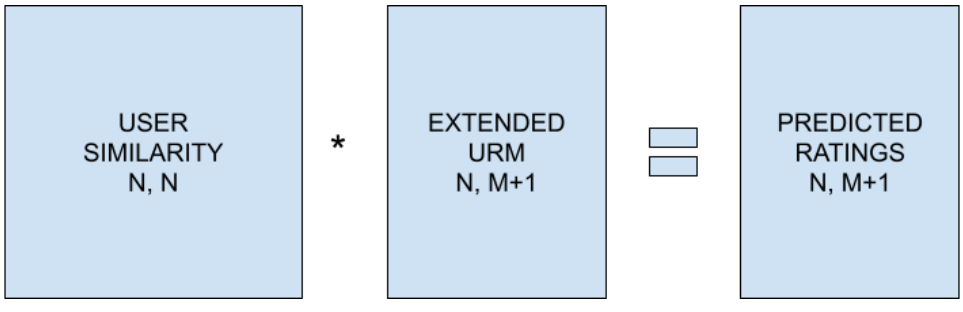


Figure 5.5: A visual representation of how the predicted ratings are calculated for User CF. Each square represents a matrix. The predicted ratings are the dot product of the similarity and the extended URM with binge-watching features.

Again, we remove the $M+1$ column from the predicted ratings and proceed with the recommendation phase as described for User CF with binge-watching and binge-worthy features in 5.2.1.

User Collaborative Filtering with binge-worthy features

Instead, using only the binge-worthy features means the extended URM has shape $N+1, M$, and the similarity matrix will have size $N+1, N+1$. The predicted rating matrix has size $N+1, M$ then.

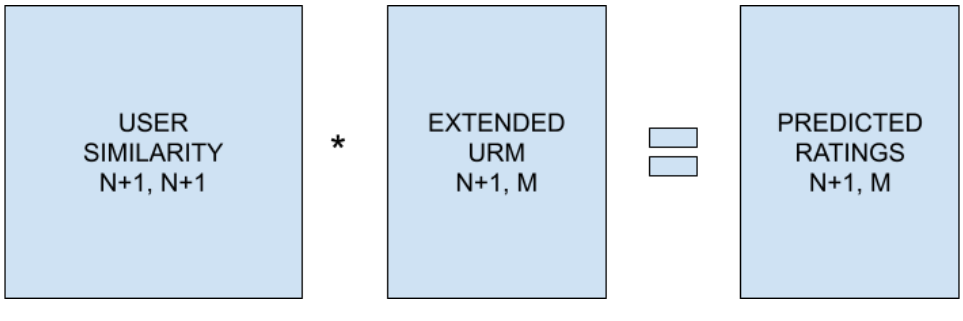


Figure 5.6: A visual representation of how the predicted ratings are calculated for User CF. Each square represents a matrix. The predicted ratings are the dot product of the similarity and the extended URM with binge-worthy features.

We have no fictitious item since we have binge-worthy features only: no pruning of the predicted rating matrix is required. We sort the items in a

decreasing order accordingly to the ratings and recommend the top-k ranked ones.

5.2.2 Item Collaborative Filtering

Item Collaborative Filtering with binge-watching and binge-worthy features

The extended URM with binge-watching and binge-worthy features is fed into an Item Collaborative Filtering model. It computes the item similarity as described in 2.1.3 using the extended URM instead of the normal one: the item similarity matrix has shape $M + 1, M + 1$, where the extra item is the binge-watcher feature array. To compute the predicted rating matrix, we compute the dot product between the item similarity and the extended URM. The predicted rating matrix has the same size as the extended URM, $N + 1, M + 1$.

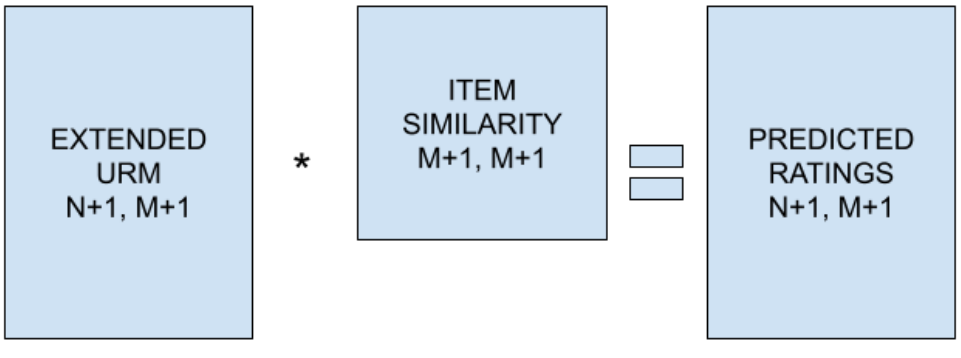


Figure 5.7: A visual representation of how the predicted ratings are calculated for Item CF. Each square represents a matrix. The predicted ratings are the dot product of the similarity and the extended URM with binge-watching and binge-worthy features.

Similarly to the User CF methods, we do not want the last item representing the binge-watching feature array to be recommended, so we remove the $M + 1$ column from the predicted ratings.

Again, for each user u of the predicted rating matrix, we sort the items in a decreasing order accordingly to the ratings. We then recommend the top-k ranked items.

The fictitious $M + 1$ user's scores corresponding to the binge-worthy features will still not be relevant during evaluation.

Item Collaborative Filtering with binge-watching features

Using only the binge-watching features leads to an extended URM with size $N, M+1$, the item similarity matrix accordingly has size $M+1, M+1$. The predicted rating matrix has size $N, M+1$ then.

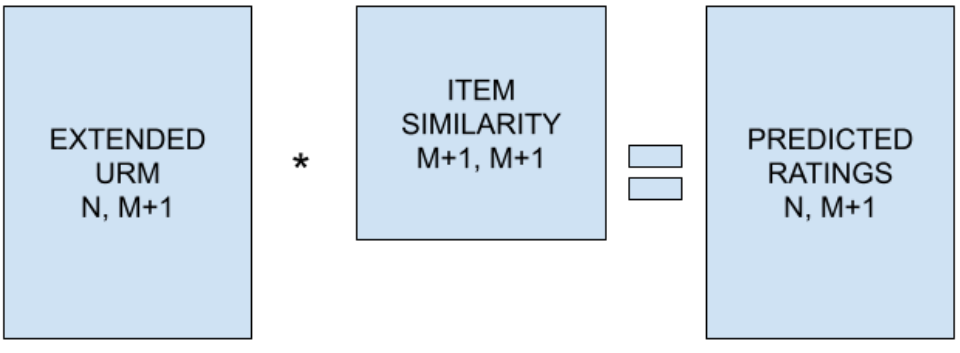


Figure 5.8: A visual representation of how the predicted ratings are calculated for Item CF. Each square represents a matrix. The predicted ratings are the dot product of the similarity and the extended URM with binge-watching features.

Again, we remove the $M+1$ column from the predicted ratings and proceed to the recommendation phase as described for Item CF with binge-watching and binge-worthy features in 5.2.2.

Item Collaborative Filtering with binge-worthy features

Finally, using only the binge-worthy features means the extended URM has shape $N+1, M$. The item similarity matrix will have the exact size of the one described in 2.1.3, M, M. The predicted rating matrix has size $N+1, M$ consequently.

Like User CF with binge-worthy features, we have no fictitious item: the predicted rating matrix does not need any adjustment, as this extra item will ever be recommended. We sort the items in a decreasing order accordingly to the ratings and recommend the top-k ranked ones.

5.2.3 SLIM with contextual information

We present a SLIM model with contextual information to include binge-watching and binge-worthy features. SLIM with contextual information is a SLIM model which takes as input the extended URM. The similarity matrix W computed by SLIM is the same one obtained in Equation 2.5b, solving an optimization problem, and it will have size $M+1, M+1$ if the extended

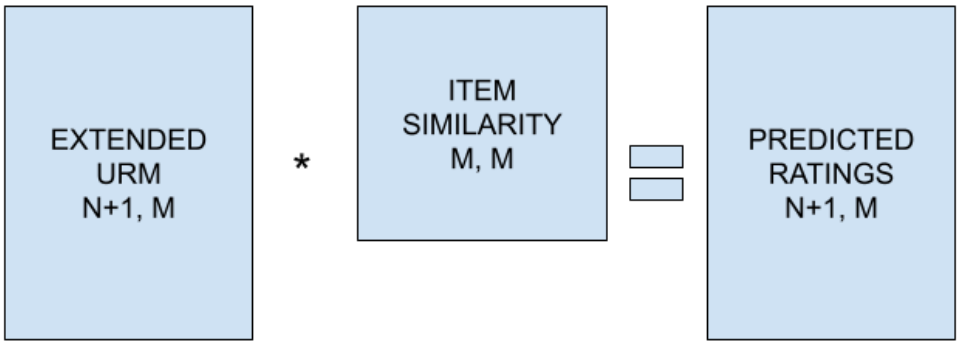


Figure 5.9: A visual representation of how the predicted ratings are calculated for Item CF. Each square represents a matrix. The predicted ratings are the dot product of the similarity and the extended URM with binge-worthy features.

URM contains binge-watching features, alone or alongside the binge-worthy ones.

Besides how the similarity is computed, SLIM with contextual information behaves like the Item CF with the extended URM presented in Section 5.2.2. The predicted rating matrix is built in the same way shown in Figure 5.7, but it uses its own computed similarity matrix W. Therefore, if we have binge-watching features in the URM, we need to remove the extra $M + 1$ item from the predicted rating matrix. Otherwise, we directly rank each user's ratings and take the top-k ones.

5.3 LightFM Factorization Machine

We introduced binge-watching and binge-worthy features in LightFM, an implementation of Factorization Machines [1] [10]. The model lets us define user and item features. It computes the latent representation of users and items as the sum of the features' latent vector, then uses it in a Matrix Factorization fashion to computer recommendations [10].

$$\hat{r}_{u,i} = f(q_u \cdot p_i + b_u + b_i)$$

$\hat{r}_{u,i}$ is the predicted rating of user u for item i, q_u is the latent representation of u, p_i is the representation of i, b_u and b_i are respectively user and item feature bias. The function $f(\cdot)$ used in our implementation is the identity function, the default function of the library.

As for the models described previously, we tested these models using binge-watching or binge-worthy features and combining both. Notably, we

[1] https://github.com/lyst/lightfm

do not need to extend the URM matrix because we can pre-process the feature arrays with built-in tools in the LightFM library.

Chapter 6

Experimental setup

In this chapter, we describe the environment in which we run tests.

In the interaction dataset, we observe the interactions between users and the catalog items, as described through Chapter 3. In particular, all items are described with a *series id* and an *item id*: movies and documentaries have a *series id* that coincides with the *item id* and are considered as series with a unique item, even though they have different item types. TV-series instead have a unique *series id* and multiple *item ids*, each referring to a different episode. We chose to recommend *series ids* instead of *item ids* because the extracted binge-watching and binge-worthy features are related to *series ids*. We recommend movies, documentaries, and TV-series through this approach, but we are not recommending single episodes.

6.1 Dataset processing

In Section 2.1.1 we summed up the main strategies to model user feedback and build URMs: implicit and explicit feedback. Classic implicit feedback approaches would assign 1 in the URM when the user interacted with the series and 0 otherwise. It does not capture the nature of user-series interaction since we would be giving the same score to a user who watched just one episode of a series and to one who saw all the episodes of a series twice. Explicit feedback does not make our case either since it builds the URM according to user-series ratings.

As previously stated, our goal is to recommend series. We create an implicit URM that counts the number of interactions each user had with a series, effectively capturing the preferences of the users with series. The more the user watches a series, the higher the number of interactions.

6.1.1 User temporal splitting

Following previous works in the area [5], we split the dataset into three different splits: train, validation, and test. After we split the dataset, we build three different URMs, one for each set.

We filter the dataset based on the filtering approaches shown in Section 3.2. Moreover, we make sure to include not only serial content, such as TV-series, but movies and documentaries as well.

We adopt a **user temporal splitting**, i.e., splitting the dataset temporally for each user. The splitting process begins by ordering the users' interaction by their timestamp in ascending order. Then, for each user, we take the first 70% of their interactions as the training set, the next 10% as the validation set, and the last 20% for the test set.

The corresponding URMs to each set are built as described in Section 6.1, but considering only the interactions of the considered set: for each pair (user, series) of the set, the URM contains the number of interactions of the user for the series within that set.

6.1.2 URM splits description

We present statistics over training, validation, and test set in Table 6.1. We show the number of interactions, user and items in each one.

	Interactions	**Users**	**Items**
Original dataset	10.457.810	42.153	28.877
Vision factor $\geq$ 90%, view interactions	4.480.691 (42.84%)	41.120 (97.55%)	20.479 (70.92%)
Training	3.117.301 (29.81%)	38.875 (92.22%)	18.279 (63.30%)
Validation	448.586 (4.29%)	35.129 (83.33%)	9.251 (32.04%)
Test	905.440 (8.66%)	38.821 (92.10%)	11.302 (39.14%)

Table 6.1: Splits statistics, in terms of the absolute number of interactions, users and series, and percentages with respect to the original unfiltered dataset. The items presented correspond to the unique $series\ ids$ present in each split. They do not represent single episodes but entire series, movies, and documentaries.

We are in a setting where we are measuring if including binge-watching and binge-worthy information generates improvements. Also, we want to evaluate the recommenders in the most realistic scenario possible, where the user has been interacting with a series for quite some time, and their

interactions can be in the three dataset splits. Moreover, recommending a series again is not considered a mistake because we want to see if engagement with it is reached.

6.1.3 Binge-watching and binge-worthy features built with training set

In Section 4.2.4 we showed statistics over a filtered dataset to identify binge-watchers and binge-worthy series. Coherently to the experimental splits described in Section 6.1.1, we computed the binge-watching and binge-worthy features array on the training set data, in the same way we described in Section 4.3.

	Min	**Max**	**Mean**	**Variance**
Binge-watching	0	1	0.11490463	0.05384468
Binge-watching > 0	0.01408	1	0.4235	0.06778
Binge-worthy	0	1	0.06271	0.03220
Binge-worthy > 0	0.01370	1	0.46184	0.05282

Table 6.2: Minimum, maximum, average value, and variance for binge-watching and binge-worthy feature arrays built over the training set. In the second and the fourth rows, we show statistics considering only the features values that are > 0.

6.2 Models training

6.2.1 Evaluation procedure

In Figure 6.1, we show the evaluation process to train recommenders. The approach is the following: We start from the filtered dataset and create the URM splits, as mentioned in Section 6.1, then, we tune the hyper-parameters of each recommender in our study by training them with the train split and evaluating them against the validation split. After the hyper-parameter tuning process finishes, we train each recommender with the best set of hyper-parameters and with the sum of the train and validation split. Then it is evaluated against the test split.

6.2.2 Models

Starting from the code from the library[1] of ContentWise Impression dataset reference article [15], we tested the following Recommender System models as baselines:

[1] https://github.com/ContentWise/contentwise-impressions

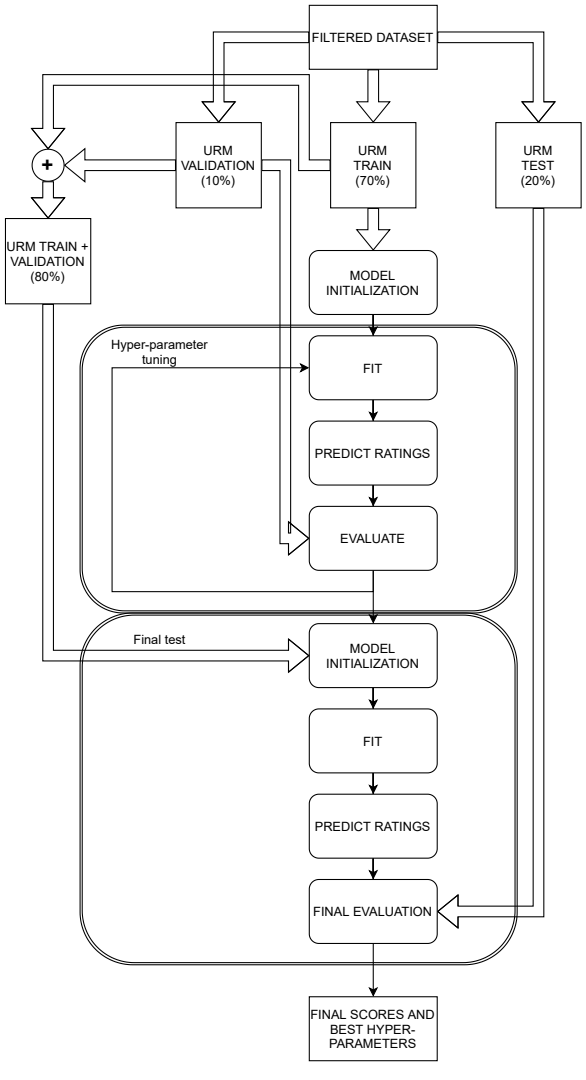


Figure 6.1: Evaluation procedure description. We initialize the model with the URM train, then we perform the hyper-parameter tuning as an iteration of fit, prediction and evaluation against the URM validation. Finally, we fit the models with the best hyper-parameters and we compute predictions with the sum of URM train and validation, evaluating the scores against the URM Test.

- Random;
- Top Popular;
- User Collaborative Filtering;

- Item Collaborative Filtering;
- SLIM ElasticNet;
- LightFM Factorization Machine [10].

We extended User Collaborative Filtering, Item Collaborative Filtering, SLIM ElasticNet, and LightFM with binge-watching and binge-worthy features, as described through Chapter 6. In particular, we weighted the features with an *alpha-weight* approach, multiplying the feature arrays for a hyper-parameter α, ranging from 0 to 1, learned during training. For each of these models, we built three different versions:

- one using binge-watching features only;
- one using binge-worthy features only;
- one combining both binge-watching and binge-worthy features.

Considering how we introduce features into SLIM ElasticNet with contextual information, it cannot be considered a pure Context-aware Recommender System since we do not define any feature matrix[22]. On the other hand, the core idea is the same since we are still introducing contextual information.

6.2.3 Keeping already seen items in recommendations

For the same reasons previously presented, we recommend already seen items, i.e., series present in the training set. We already know that an element present in the training set will not be in the validation or test set in recommenders without overlapping sets. However, in our case, we want to recommend the missing episodes of a series to a user who might not have finished in the observation span in which we gather the interactions in the training set. It makes sense to recommend series that are already present in the dataset. Considering how the URMs are built, we recommend the number of interactions we think a user would have with a particular series.

6.2.4 Hyper-parameter tuning

We tuned the reported models' hyper-parameters, optimizing the recommendation performance over the validation set. We used the Bayesian Optimization implemented in the reference library [15], optimizing the MAP metric. We trained each model for 50 iterations, with 15 random starts each. During the first 15 epochs, the optimizer explores random parameters and

then exploits the best one over the validation set. At the end of the tuning, we train the recommender using the best parameters found using the union of the training and the validation set and scoring it against the test set.

6.2.5 Metrics

We evaluated the performance of top-N recommendations over the following metrics: Precision (PREC), Recall (REC), Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), F1, Novelty, Coverage, and Diversity. Each metric is computed with cut-off values 5, 10 and 20, which means considering the $@K$ metrics as described in Section 2.3.

Chapter 7

Results

In this chapter, we present an analysis of the performance of models trained using binge-watching and binge-worthy features. We present results grouped by model type to show how they compare with and without the features. We show MAP and Recall metrics, with cut-offs of 5, 10, and 20. We also present other accuracy and beyond-accuracy metrics with a cut-off of 20.

7.1 Random and Top Popular baselines

In Tables 7.1 and 7.2, we show the results for two baselines to illustrate the difference in accuracy between them and more advanced models trained with and without features. Random model recommends the series randomly, which explains its low accuracy but high diversity and coverage. Instead, Top Popular recommends the first k most popular items, where k is the considered cut-off. On the contrary, this recommender has a higher value of accuracy, but its diversity is zero as recommendations are not personalized.

	@ 5		@ 10		@ 20	
	REC	**MAP**	**REC**	**MAP**	**REC**	**MAP**
Random	0.0001	0.0001	0.0002	0.0001	0.0003	0.0001
TopPop	0.126	0.0988	0.1614	0.0998	0.1975	0.1016

Table 7.1: Baselines for Random and Top Popular recommenders. They do not use features.

	@ 20									
	PREC	**MRR**	**F1**	**ARHR**	**Novelty**	**Div.MIL**	**Div.HHI**	**Cov. Item**	**Div. Gini**	**Div. Shannon**
Random	0.0001	0.0003	0.0001	0.0003	0.0182	0.9989	0.9999	1	0.909	14.139
TopPop	0.0299	0.1851	0.052	0.2199	0.0083	0	0.95	0.0011	0.0011	4.3219

Table 7.2: Baselines with all accuracy measures for Random and Top Popular recommenders. They do not use features.

7.2 Collaborative Filtering

7.2.1 User Collaborative Filtering

We outline the results for User Collaborative Filtering using the extended URM. We group the results by the similarity measure.

			@ 5		@ 10		@ 20	
	Weight binge-watchers	**Weight binge-worthy**	**REC**	**MAP**	**REC**	**MAP**	**REC**	**MAP**
UserKNN CF cosine	-	-	0.4707	0.3763	0.5489	0.3805	0.6181	0.3886
Bingewatcher Bingeworthy	0	0	0.4431	0.3641	0.5081	0.3648	0.5668	0.3706
Bingewatcher	0	-	0.4431	0.3641	0.508	0.3648	0.5668	0.3706
Bingeworthy	-	0.0574	0.4431	0.3641	0.5081	0.3648	0.5669	0.3706
UserKNN CF dice	-	-	0.3677	0.2699	0.4514	0.2788	0.532	0.2878
Bingewatcher Bingeworthy	0	1	0.3667	0.2693	0.4504	0.2781	0.5313	0.2872
Bingewatcher	0	-	0.368	0.2695	0.452	0.2785	0.5322	0.2876
Bingeworthy	-	0	0.3675	0.2697	0.4512	0.2785	0.5319	0.2876
UserKNN CF jaccard		-	0.3688	0.2706	0.453	0.2795	0.5328	0.2886
Bingewatcher Bingeworthy	0	0	0.3683	0.2701	0.4517	0.2788	0.5322	0.2879
Bingewatcher	0	-	0.3681	0.2702	0.4518	0.2789	0.532	0.288
Bingeworthy	-	0	0.3687	0.2706	0.4523	0.2794	0.5332	0.2885
UserKNN CF asymmetric	-	-	0.4641	0.3724	0.5423	0.3763	0.6117	0.3842
Bingewatcher Bingeworthy	0	0	0.4431	0.3641	0.5081	0.3648	0.5668	0.3706
Bingewatcher	0	-	0.4431	0.3641	0.508	0.3648	0.5668	0.3706
Bingeworthy	-	0	0.4431	0.3641	0.5081	0.3648	0.5668	0.3706
UserKNN CF tversky	-	-	0.387	0.2848	0.4718	0.2934	0.5472	0.3022
Bingewatcher Bingeworthy	0	0	0.3689	0.2701	0.4527	0.279	0.533	0.2882
Bingewatcher	0	-	0.3688	0.2707	0.4529	0.2795	0.5327	0.2886
Bingeworthy	-	0	0.3688	0.2705	0.4523	0.2793	0.5333	0.2885

Table 7.3: Compared results for MAP and Recall between User Collaborative filtering for each similarity computation. Results are shown in chunks for each similarity without features, and with all possible feature combinations

Extending the URM of a User Collaborative Filtering with binge-watching or binge-worthy features does not improve its performances. The information introduced is not leveraged by this model.

In Table 7.4 we show the results for other accuracy and beyond-accuracy metrics at cut-off 20. All metrics are impacted negatively by binge-watching and binge-worthy features, even when their weights are set to 0. The

	@ 20								
	PREC	**MRR**	**F1**	**Novelty**	**Div. MIL**	**Div. HHI**	**Cov. Item**	**Div. Gini**	**Div. Shannon**
UserKNN CF cosine	0.0845	0.5448	0.1487	0.0104	0.857	0.9928	0.2449	0.0234	8.6923
Bingewatcher Bingeworthy	0.0739	0.5392	0.1307	0.0104	0.8278	0.9914	0.2372	0.0199	8.3794
Bingewatcher	0.0739	0.5392	0.1307	0.0104	0.8278	0.9914	0.2372	0.0199	8.3794
Bingeworthy	0.0739	0.5392	0.1307	0.0104	0.8278	0.9914	0.2372	0.0199	8.3795
UserKNN CF dice	0.0692	0.4088	0.1225	0.0108	0.8321	0.9916	0.199	0.0185	8.3596
Bingewatcher Bingeworthy	0.0691	0.4084	0.1223	0.0108	0.8266	0.9913	0.1953	0.0178	8.3033
Bingewatcher	0.0692	0.4075	0.1225	0.0108	0.8387	0.9919	0.205	0.0195	8.435
Bingeworthy	0.0692	0.4086	0.1225	0.0108	0.8311	0.9916	0.1982	0.0183	8.3487
UserKNN CF jaccard	0.0694	0.4093	0.1228	0.0108	0.8327	0.9916	0.2021	0.0187	8.3727
Bingewatcher Bingeworthy	0.0693	0.409	0.1227	0.0108	0.8286	0.9914	0.1991	0.0182	8.3278
Bingewatcher	0.0693	0.4092	0.1226	0.0107	0.8274	0.9914	0.1982	0.018	8.315
Bingeworthy	0.0694	0.4092	0.1228	0.0108	0.8352	0.9918	0.2038	0.0191	8.4
UserKNN CF asymmetric	0.0837	0.543	0.1473	0.0103	0.8425	0.9921	0.2309	0.0203	8.4849
Bingewatcher Bingeworthy	0.0739	0.5392	0.1307	0.0104	0.8278	0.9914	0.2372	0.0199	8.3794
Bingewatcher	0.0739	0.5392	0.1307	0.0104	0.8278	0.9914	0.2372	0.0199	8.3794
Bingeworthy	0.0739	0.5392	0.1307	0.0104	0.8278	0.9914	0.2372	0.0199	8.3794
UserKNN CF tversky	0.0721	0.4287	0.1274	0.0108	0.8416	0.9921	0.1962	0.0184	8.4059
Bingewatcher Bingeworthy	0.0694	0.4081	0.1228	0.0108	0.8383	0.9919	0.2068	0.0196	8.4364
Bingewatcher	0.0694	0.4094	0.1228	0.0108	0.8326	0.9916	0.202	0.0187	8.3715
Bingeworthy	0.0694	0.4091	0.1228	0.0108	0.8353	0.9918	0.2039	0.0191	8.4014

Table 7.4: Compared results for other accuracy metrics between User Collaborative filtering for each similarity computation. Results are shown in chunks for each similarity without features, and with all possible feature combinations. The features weights are omitted for readability reasons. They are the same of the ones shown in Table 7.3.

Bayesian optimization process, in fact, selected not to use these features. This means the best results obtained are the ones where features' weights are not considered. Even when it uses them, the weights are close to 0, and the results are not impacted, as shown for UserKNN CF cosine with binge-worthy features in Table 7.3. Hence, we can conclude that User Collaborative Filtering is not impacted by this information.

7.2.2 Item Collaborative Filtering

In Table 7.5 we indicate the results for Item Collaborative Filtering using the extended URM. The results are still grouped by similarity measure.

Considering the general pattern among the different models trained, we cannot conclude that Item Collaborative Filtering is improved through binge-watching and binge-worthy, even though we can improve by a 1.16% margin the MAP@20, and by a similar amount the other metrics. This margin is not sufficiently wide to consider this an improvement. The Bayesian optimization process selects to use binge-worthy features. In most cases, the model's performances drop significantly when using features, and where there is an improvement, it is marginal.

Moreover, we notice that all the other metrics, including the beyond-

			@ 5		@ 10		@ 20	
	Weight binge-watchers	**Weight binge-worthy**	**REC**	**MAP**	**REC**	**MAP**	**REC**	**MAP**
ItemKNN CF cosine	-	-	0.1077	0.0709	0.1785	0.0769	0.2698	0.0858
Bingewatcher Bingeworthy	1	0	0.0746	0.0522	0.1235	0.0555	0.1929	0.0621
Bingewatcher	1	-	0.0746	0.0522	0.1235	0.0555	0.1929	0.0621
Bingeworthy	-	0	0.0742	0.0519	0.1223	0.0552	0.1912	0.0616
ItemKNN CF dice	-	-	0.1084	0.0693	0.1847	0.0763	0.277	0.0856
Bingewatcher Bingeworthy	1	0	0.1098	0.069	0.1848	0.0757	0.276	0.0845
Bingewatcher	1	-	0.11	0.0691	0.1851	0.0758	0.2767	0.0847
Bingeworthy	-	0	0.1084	0.0693	0.1847	0.0763	0.277	0.0856
ItemKNN CF jaccard	-	-	0.1069	0.0692	0.1854	0.0766	0.2828	0.0863
Bingewatcher Bingeworthy	0	0	0.1075	0.0692	0.1817	0.0757	0.2714	0.0845
Bingewatcher	**0.91**	-	**0.1094**	**0.0701**	**0.1883**	**0.0776**	**0.2856**	**0.0873**
Bingeworthy	-	0	0.1075	0.0693	0.1816	0.0757	0.2715	0.0846
ItemKNN CF asymmetric	-	-	0.1785	0.1258	0.2594	0.1315	0.3591	0.1416
Bingewatcher Bingeworthy	1	0	0.0747	0.0521	0.1236	0.0555	0.1929	0.062
Bingewatcher	1	-	0.0746	0.0521	0.1236	0.0555	0.193	0.0621
Bingeworthy	-	0	0.0742	0.0519	0.1224	0.0552	0.1912	0.0616
ItemKNN CF tversky	-	-	0.1268	0.0885	0.1947	0.0929	0.2832	0.1019
Bingewatcher Bingeworthy	1	0	0.1125	0.0706	0.1876	0.0772	0.2783	0.0861
Bingewatcher	1	-	0.1157	0.0722	0.1921	0.0792	0.284	0.0883
Bingeworthy	-	0.1387	0.1041	0.0677	0.1816	0.0747	0.277	0.084

Table 7.5: Compared results between Item Collaborative filtering for each similarity computation. Results are shown in chunks for each similarity without features and all possible feature combinations. We highlight in bold the rows where we were able to improve the results for the corresponding baseline

accuracy ones, are generally impacted negatively by binge-watching or binge-worthy features. We show those results in Table 7.6.

The results of Item Collaborative Filtering are similar to those of User Collaborative Filtering, i.e., these two types of algorithms do not leverage binge-watching or binge-worthy features. We report these results for completeness.

7.3 SLIM with contextual information

In Table 7.7 we outline the results provided by SLIM ElasticNet with contextual information, both with and without features. We introduce features by extending the URM.

Using both information on binge-watchers and binge-worthy series, we improve the model by 5.39% approximately, in terms of MAP@20. Concerning Recall, we obtain more minor improvements. We outline that the improvements in MAP and Recall obtained by using binge-watching and binge-worthy features correspond to improvements in every other metric, as shown in Table 7.8.

	@ 20								
	PREC	**MRR**	**F1**	**Novelty**	**Div. MIL**	**Div. HHI**	**Cov. Item**	**Div. Gini**	**Div. Shannon**
ItemKNN CF cosine	0.0501	0.1627	0.0845	0.012	0.9298	0.9965	0.772	0.0903	10.1542
Bingewatcher Bingeworthy	0.0395	0.1301	0.0656	0.0116	0.8386	0.9919	0.3616	0.0523	9.184
Bingewatcher	0.0395	0.1301	0.0656	0.0116	0.8386	0.9919	0.3616	0.0523	9.184
Bingeworthy	0.0393	0.1295	0.0652	0.0117	0.8455	0.9923	0.3631	0.0535	9.247
ItemKNN CF dice	0.0489	0.1629	0.0832	0.0121	0.9381	0.9969	0.8058	0.0856	10.2387
Bingewatcher Bingeworthy	0.0485	0.1631	0.0825	0.0116	0.9279	0.9964	0.5658	0.0613	9.9317
Bingewatcher	0.0486	0.1633	0.0826	0.0116	0.9293	0.9965	0.5734	0.0625	9.9601
Bingeworthy	0.0489	0.1629	0.0832	0.0121	0.9381	0.9969	0.8058	0.0856	10.2387
ItemKNN CF jaccard	0.0498	0.1636	0.0847	0.0118	0.9259	0.9963	0.6958	0.0731	10.035
Bingewatcher Bingeworthy	0.0486	0.1632	0.0824	0.0117	0.9332	0.9967	0.5701	0.0621	9.9761
Bingewatcher	0.0499	0.1654	0.085	0.0117	0.9244	0.9962	0.6825	0.0714	10.0032
Bingeworthy	0.0486	0.1633	0.0824	0.0117	0.9333	0.9967	0.5704	0.0623	9.9793
ItemKNN CF asymmetric	0.0602	0.2567	0.1031	0.0105	0.8476	0.9924	0.2917	0.0246	8.6179
Bingewatcher Bingeworthy	0.0395	0.13	0.0655	0.0116	0.8417	0.9921	0.363	0.0529	9.213
Bingewatcher	0.0395	0.1301	0.0655	0.0116	0.8414	0.9921	0.3629	0.0528	9.2104
Bingeworthy	0.0393	0.1294	0.0652	0.0117	0.8459	0.9923	0.3633	0.0536	9.2499
ItemKNN CF tversky	0.0511	0.1979	0.0866	0.011	0.9256	0.9963	0.4392	0.0356	9.3438
Bingewatcher Bingeworthy	0.0491	0.1654	0.0834	0.0115	0.9254	0.9963	0.5606	0.0595	9.8811
Bingewatcher	0.0499	0.168	0.0849	0.0118	0.9304	0.9965	0.6004	0.0662	10.0119
Bingeworthy	0.0491	0.1609	0.0834	0.0115	0.9226	0.9961	0.5786	0.0613	9.876

Table 7.6: Compared results for other accuracy metrics between Item Collaborative filtering for each similarity computation. Results are shown in chunks for each similarity without features, and with all possible feature combinations. The features weights are omitted for readability reasons. They are the same of the ones shown in Table 7.5

			@ 5		@ 10		@ 20	
	Weight binge-watchers	**Weight binge-worthy**	**REC**	**MAP**	**REC**	**MAP**	**REC**	**MAP**
SLIM ElasticNet	-	-	0.1383	0.0925	0.2078	0.0974	0.2951	0.1058
Bingewatcher Bingeworthy	**0.7437**	**1**	**0.1452**	**0.0979**	**0.2163**	**0.1029**	**0.3042**	**0.1115**
Bingewatcher	0.6356	-	0.1364	0.091	0.2066	0.096	0.2929	0.1044
Bingeworthy	-	**0.2665**	**0.1379**	**0.0937**	0.2063	**0.098**	0.2917	**0.1063**

Table 7.7: Results for SLIM ElasticNet models. Results are shown without features and with all possible feature combinations. We highlight in bold the improved metrics for the baseline and the corresponding weights

	@ 20								
	PREC	**MRR**	**F1**	**Novelty**	**Div. MIL**	**Div. HHI**	**Cov. Item**	**Div. Gini**	**Div. Shannon**
SLIM ElasticNet	0.0502	0.1993	0.0858	0.0113	0.875	0.9938	0.2244	0.0255	8.8454
Bingewatcher Bingeworthy	0.0513	0.2076	0.0878	0.0114	0.8925	0.9946	0.2636	0.0309	9.12
Bingewatcher	0.0498	0.1977	0.0852	0.0112	0.8756	0.9938	0.2205	0.0248	8.8202
Bingeworthy	0.0505	0.2016	0.0861	0.0115	0.9049	0.9952	0.2607	0.0311	9.1693

Table 7.8: Results for other accuracy metrics for SLIM ElasticNet models. Results are shown in chunks for each similarity without features, and with all possible feature combinations. The features weights are omitted for readability reasons. They are the same of the ones shown in Table 7.7

In contrast to User and Item Collaborative Filtering techniques, SLIM ElasticNet leverages both the binge-watchers and binge-worthy series information, improving the recommendation quality. This can be explained by how SLIM computes the similarity matrix, which is learned following the optimization approach explained in Section 2.1.4. Moreover, this result shows how Machine Learning models can leverage this type of extra content while non-ML models cannot.

7.4 LightFM

Last, we present results for LightFM, comparing the non-feature version with the ones using features in all possible combinations.

			@ 5		@ 10		@ 20	
	Weight binge-watchers	**Weight binge-worthy**	**REC**	**MAP**	**REC**	**MAP**	**REC**	**MAP**
LightFM	-	-	0.3649	0.2253	0.5116	0.2452	0.6401	0.2618
Bingewatcher bingeworthy	0	1	**0.3744**	**0.2311**	**0.5199**	**0.2541**	**0.6442**	**0.2706**
Bingewatcher	0	-	0.3731	0.2338	0.5174	0.2526	0.6453	0.2691
Bingeworthy	-	1	**0.3773**	**0.2361**	**0.5202**	**0.2551**	**0.6446**	**0.2715**

Table 7.9: Results for LightFM models. Results are shown without features and with all possible feature combinations. We highlight in bold the improved metrics for the baseline and the corresponding weights

As shown in Section 7.3, binge-worthy features improve the recommendation accuracy. The hyper-parameter tuning process ends up reaching the maximum importance weight for binge-worthy features, though it judges binge-watching information as non-important, setting its weight to 0. LightFM with binge-watching features improves MAP@20 by 3.7%. The other metrics are impacted at a lower rate instead, as shown in Table 7.10. Notice

	@ 20								
	PREC	**MRR**	**F1**	**Novelty**	**Div. MIL**	**Div. HHI**	**Cov. Item**	**Div. Gini**	**Div. Shannon**
LightFMWrapper	0.0835	0.3631	0.1477	0.0114	0.9212	0.9961	0.9191	0.1016	10.3013
Bingewatcher Bingeworthy	0.0849	0.3737	0.1499	0.0112	0.9064	0.9953	0.8824	0.0864	10.0334
Bingewatcher	0.085	0.3754	0.1503	0.0115	0.9248	0.9962	0.9136	0.1023	10.3657
Bingeworthy	0.0852	0.3763	0.1505	0.0113	0.9111	0.9956	0.9075	0.0948	10.1532

Table 7.10: Results for other accuracy metrics for LightFM models. Results are shown in chunks for each similarity without features, and with all possible feature combinations. The features weights are omitted for readability reasons. They are the same of the ones shown in Table 7.9

that LightFM with only binge-watching information has a higher score than

the baseline. The improvement is due to the discovery of better hyper-parameters during the search, and the binge-watching features do not impact the scores since the best weight is chosen 0.

7.5 Popularity bias analysis

Table 7.11 shows how much binge-watching and binge-worthy features are correlated to the popularity of users and items, respectively. We sum the interactions of the training URM column-wise to obtain user popularity, and we do the same row-wise to obtain the series popularity.

	Pearson		**Spearman**		**Kendall**	
	r	p val	r	p val	tau	p val
Series popularity, binge-worthy features	0.0211	0.004338	0.4873	0	0.397	0
User popularity, binge-watching features	0.1951	0	0.5364	0	0.417	0

Table 7.11: Different correlation metrics measured between user popularity and binge-watching features, series popularity and binge-worthy features

As said in Section 2.5.1, Pearson-r values can go between -1 and 1, with 0 implying no correlation. Based on the values shown in Table 7.11, user and series popularities have a low linear correlation between binge-watching and binge-worthy features, respectively. Instead, they have a certain degree rank correlation due to Spearman-r and Kendall-tau's values. The definitions of binge-watcher and binge-worthy series are intrinsically related to popularity: they both count the number of binge-watching sessions for a user or for a series to attribute the binge-status. Series that are highly popular are more prone to be binge-watched, even from everyday experience.

To conclude, we are introducing new information since the correlation measures are low in terms of linear correlation and range from 0.4 to 0.5 approximately for rank correlation ones. We expect some degree of correlation between binge-features and popularity arrays. However, it is not sufficient to state information on binge-watchers and binge-worthy series is a redundant form of popularity.

7.6 Time benchmarks

In Appendix A.1, we show the time needed to train the tested models, both with and without features.

We train models using Amazon Web Services[1] EC-2 machines. Mostly,

[1]https://aws.amazon.com/

we use a c5.4xlarge instance with 16vCPUs, 32 GB of memory, and running Amazon Linux version 2.

7.7 Final remarks

To conclude, KNN models exploit neither the binge-watching nor binge-worthy features. Instead, our implementation of SLIM ElasticNet using the extended URM, similarly to a Context-aware model, exploits both binge-worthy and binge-watching information, though binge-worthy is dominant.

LightFM with binge-worthy features is improved by 3.7% in terms of MAP@20. Binge-worthy features are weighted at the maximum value. Instead, binge-watching features are discarded during optimization.

Finally, binge-worthy information appears to be relevant for Factorization Machines and SLIM ElasticNet model because it improves recommendations. Moreover, these features are not strongly correlated to series popularity; hence the information we are introducing is primarily new. On the other hand, binge-watching features do not improve the Recommender Systems for which we have provided results.

Chapter 8

Conclusion and Future Work

This chapter presents and discusses the key outputs and contributions of our research work. This work aims at structuring the process of binge-watchers identification on an industrial dataset, translating the research in other fields that analyzes and defines binge-watching behavior from a different perspective, often too general or relying solely on surveys.

The absence of well-defined approaches to identify binge-watchers and binge-worthy series in a dataset leads to a consequent non-existent exploration of its significance. Currently, the research community does not provide any hints on whether this behavioral pattern improves recommendation quality or not. The second goal of this thesis is to track how the extracted information impacts Recommender Systems, hence understanding if it can improve their performance.

8.1 Outputs and Contributions

We divide our work into two main parts: identifying binge-watchers and binge-worthy series in our dataset and analyzing the results from models built to include these features.

For the first task, we extract binge-watchers and binge-worthy series from a filtered dataset with only serial content and relevant interactions by computing watching sessions and applying the binge-watching definitions we find in the literature. We then compare the statistics over binge-watchers that can be found in surveys and current research and validate our approach's efficacy by matching those statistics accordingly. Finally, we translate binge-watchers and binge-worthy series information into features that will be plugged into models. To summarize, we provide the reader with a flexible set of tools to extract binge-watching information from a serial

content dataset.

Secondly, we propose different strategies to introduce these features into models. The first approach was to extend the User Rating Matrix using a fake item corresponding to the binge-watchers information and a fake user corresponding to the binge-worthy series features. The second one was to plug these features into LightFM, a Factorization Machine implementation that digests well user and item features and uses them to make recommendations. The two most improved algorithms are SLIM ElasticNet with contextual information, where the context is introduced by extending the URM, and LightFM. In both cases, we verify that binge-worthy features improve the recommendation quality. Concerning binge-watching features, they do not seem to improve the recommendations.

8.2 Limitations

Apart from the multiple contributions brought to this research, limitations are still present and need to be assessed, as it is going to be discussed throughout this section.

Due to the computational power required to train models and score prediction, we must limit the number of epochs in which the hyper-parameter tuning runs.

Secondly, most industrial datasets are private, unlike the one we use. It would be interesting to validate our approaches on the datasets used in the previous articles and surveys.

Finally, there are no recommendation algorithms available in the literature that use this type of information to compare our results due to the scarcity of research in this field, so we must rely only on baselines.

8.3 Future Work

Research on binge-watching features in Computer Science-related fields is almost non-existent; hence, future research must be performed, particularly in terms of using the features extracted.

Even though binge-watching information is not relevant to the recommendation models from our experiments and analysis, it still can be used as metadata to enrich user profiles. For instance, knowing that a user is a binge-watcher could be used to arrange specifically the carousels of recommended items within an OTT media service page. We could order the items inside the carousel using binge-worthy information that we have proven to

be efficient in item recommendations. These ideas can be validated with A/B testing after being deployed in production.

Bibliography

[1] George Anghelcev, Sela Sar, Justin Martin, and Jas Moultrie. Binge-watching serial video content: Exploring the subjective phenomenology of the binge-watching experience. the role of narrative transportation. *Mass Communication & Society*, 24:130–154, 01 2021.

[2] Lee Raymond Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945.

[3] DTVE Reporter. Binge viewing the 'new norm', says TiVo, July 2015. publisher: Digital TV Europe.

[4] Jean Mark Gawron. Improving sparse word similarity models with asymmetric measures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 296–301, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[5] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[6] Paul Jaccard. The distribution of the flora in the alpine zone.1. *New Phytologist*, 11(2):37–50, 1912.

[7] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 47–51, New York, NY, USA, 2010. Association for Computing Machinery.

[8] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[9] William H. Kruskal. Ordinal measures of association. *Journal of the American Statistical Association*, 53(284):814–861, 1958.

[10] Maciej Kula. Metadata embeddings for user and item cold-start recommendations, 2015.

[11] J. L. Myers and A. D. Well. *Research Design and Statistical Analysis*. Lawrence Erlbaum Associates, New Jersey, 2003.

[12] Netflix, Inc. Netflix declares binge watching is the new normal, December 2013. publisher: PRNewswire.

[13] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM 11, pages 497–506, USA, 2011. IEEE Computer Society.

[14] Karl Pearson. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London Series I*, 58:240–242, January 1895.

[15] Fernando B. Pérez Maurera, Maurizio Ferrari Dacrema, Lorenzo Saule, Mario Scriminaci, and Paolo Cremonesi. Contentwise impressions: An industrial dataset with impressions included. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, pages 3093–3100, New York, NY, USA, 2020. Association for Computing Machinery.

[16] L.G. Perks. *Media Marathoning: Immersions in Morality*. Lexington Books, 2014.

[17] S. Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[18] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*, volume 1-35, pages 1–35. 10 2010.

[19] Lorenzo Saule. Negative item sampling on user impressions for multichannel bayesian personalized ranking techniques. Master's thesis, Politecnico di Milano, April 2020.

[20] William Trouleau, Azin Ashkan, Weicong Ding, and Brian Eriksson. Just one more: Modeling binge watching behavior. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery*

and Data Mining, KDD '16, page 1215 to 1224, New York, NY, USA, 2016. Association for Computing Machinery.

[21] Amos Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[22] Yong Zheng, Bamshad Mobasher, and Robin Burke. CSLIM: Contextual slim recommendation algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 301–304, New York, NY, USA, 2014. Association for Computing Machinery.

Appendix A

Results

A.1 Time benchmarks

	Train Time	RecommendationTime	Recommendation Throughput
Random	0.01 [sec]	36.93 [sec]	1055
TopPop	0.01 [sec]	43.79 [sec]	890
UserKNN CF cosine	20.39 ± 2.90 [sec]	197.74 [sec] / 3.30 ± 0.09 [min]	191
Bingewatcher Bingeworthy	27.12 ± 1.08 [sec]	182.20 [sec] / 3.04 ± 0.01 [min]	214
Bingewatcher	25.96 ± 1.26 [sec]	178.38 [sec] / 2.97 ± 0.00 [min]	219
Bingeworthy	36.71 ± 5.78 [sec]	247.15 [sec] / 4.12 ± 0.56 [min]	146
UserKNN CF dice	16.19 ± 0.90 [sec]	155.13 [sec] / 2.59 ± 0.01 [min]	252
Bingewatcher Bingeworthy	26.74 ± 0.93 [sec]	180.59 [sec] / 3.01± 0.01 [min]	216
Bingewatcher	25.66 ± 0.97 [sec]	177.61 [sec] / 2.96 ± 0.01 [min]	220
Bingeworthy	30.30 ±5.54 [sec]	250.53 [sec] / 4.18 ± 0.90 [min]	201
UserKNN CF jaccard	16.41 ±0.90 [sec]	158.74 [sec] / 2.65 ±0.11 [min]	252
Bingewatcher Bingeworthy	26.78 ± 0.89 [sec]	181.98 [sec] / 3.03 ± 0.01 [min]	214
Bingewatcher	25.54 ±1.22 [sec]	178.63 [sec] / 2.98 ± 0.01 [min]	219
Bingeworthy	33.92 ±4.88 [sec]	236.80 [sec] / 3.95± 0.35 [min]	136
UserKNN CF asymmetric	17.09 ± 1.11 [sec]	155.60 [sec] / 2.59 ±0.01 [min]	250
Bingewatcher Bingeworthy	27.33 ± 0.97 [sec]	181.43 [sec] / 3.02 ±0.02 [min]	216
Bingewatcher	26.53 ± 1.06 [sec]	178.06 [sec] / 2.97 ± 0.01 [min]	219
Bingeworthy	35.63 ± 5.30 [sec]	281.80 [sec] / 4.70 ± 0.82 [min]	125
UserKNN CF tversky	17.44 ± 0.97 [sec]	155.20 [sec] / 2.59 ±0.01 [min]	251
Bingewatcher Bingeworthy	27.38 ± 1.05 [sec]	179.55 [sec] / 2.99 ±0.01 [min]	216
Bingewatcher	27.04 ±1.19 [sec]	177.84 [sec] / 2.96±0.01 [min]	220
Bingeworthy	28.40 ± 4.63 [sec]	213.72 [sec] / 3.56 ± 0.84 [min]	199
ItemKNN CF cosine	3.21 ± 0.59 [sec]	49.10 ± 1.97 [sec]	802
Bingewatcher Bingeworthy	13.42 ±0.23 [sec]	39.54±0.36 [sec]	978
Bingewatcher	12.74 ± 2.48 [sec]	42.70±1.64 [sec]	883
Bingeworthy	12.25 ± 2.00 [sec]	38.80 ± 0.22 [sec]	1006
ItemKNN CF dice	2.96 ± 0.47 [sec]	50.17 ± 1.17 [sec]	787
Bingewatcher Bingeworthy	13.41 ± 0.57 [sec]	38.22 ± 0.42 [sec]	1025
Bingewatcher	12.49 ± 2.49 [sec]	46.53 ± 13.58 [sec]	980
Bingeworthy	12.21 ± 1.62 [sec]	41.76 ±8.45 [sec]	1028
ItemKNN CF jaccard	2.87 ± 0.58 [sec]	48.21 ± 1.11 [sec]	801
Bingewatcher Bingeworthy	13.40 ± 0.30 [sec]	38.68 ± 0.89 [sec]	1023
Bingewatcher	12.93 ± 2.56 [sec]	54.64 ± 16.19 [sec]	596
Bingeworthy	12.67 ± 2.46 [sec]	42.88 ± 12.27 [sec]	1053
ItemKNN CF asymmetric	3.38 ± 0.48 [sec]	50.31 ± 1.12 [sec]	745
Bingewatcher Bingeworthy	13.55 ± 0.19 [sec]	39.53 ±0.49 [sec]	979
Bingewatcher	12.25 ± 2.06 [sec]	42.52 ± 1.58 [sec]	936
Bingeworthy	12.33 ± 1.87 [sec]	51.53 ± 15.49 [sec]	1006
ItemKNN CF tversky	3.27 ± 0.51 [sec]	49.49 ± 1.34 [sec]	791
Bingewatcher Bingeworthy	13.66 ± 0.25 [sec]	38.17 ± 0.81 [sec]	1021
Bingewatcher	11.98 ± 1.51 [sec]	50.29 ± 17.80 [sec]	985
Bingeworthy	12.76 ± 2.10 [sec]	37.95 ± 1.11 [sec]	1047
SLIM ElasticNet	488.60 [sec] / 8.14 ± 7.06 [min]	187.86 [sec] / 3.13±0.21 [min]	213
Bingewatcher Bingeworthy	641.94 [sec] / 10.70 ±7.93 [min]	291.52 [sec] / 4.86 ± 2.50 [min]	167
Bingewatcher	606.51 [sec] / 10.11 ±8.32 [min]	252.21 [sec] / 4.20 ± 0.65 [min]	133
Bingeworthy	483.77 [sec] / 8.06±6.16 [min]	156.65 [sec] / 2.61± 0.01 [min]	249
LightFM	52.93 ± 26.18 [sec]	179.79 [sec] / 3.00 ± 0.00[min]	217
Bingewatcher bingeworthy	105.83 [sec] / 1.76 ± 1.33 [min]	967.88 [sec] / 16.13 ± 0.12 [min]	40
Bingewatcher	52.97± 29.25 [sec]	951.24 [sec] / 15.85 ± 0.27 [min]	40
Bingeworthy	90.77 [sec] / 1.51 ± 1.53 [min]	744.33 [sec] / 12.41 ± 6.07 [min]	121

Table A.1: Time benchmarks for the trained models